GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

Bachelorarbeit

Rechengitter zur Teilchenverfolgung in DSMC-Strömungssimulationen

Computational grids for particle tracking in DSMC flow simulations

angefertigt am DLR - Institut für Aerodynamik und Strömungstechnik
von Frank Stollmeier aus Delbrück-Westenholz

Bearbeitungszeit: 4. Mai 2009 bis 10. August 2009

Betreuer/in: Dipl.-Ing. Martin Grabe

Erstgutachter/in: Prof. Dr. Andreas Dillmann

Zweitgutachter/in: Prof. Dr. Wolfgang Glatzel

Zusammenfassung

Am DLR wird derzeit an der Entwicklung eines DSMC-Programms für die Simulation verdünnter Gasströmungen in dreidimensionalen Geometrien gearbeitet. Dabei wird das Gas als eine große Anzahl von Teilchen aufgefasst, die sich durch das Rechengebiet bewegen, mit Rändern kollidieren können und untereinander stoßen.
Für diese Aufgabe wird ein Rechengitter benötigt, das aus konvexen Zellen mit möglichst isotroper Ausdehnung besteht. Die Netzdichte des Rechengitters sollte sowohl lokal als auch zeitlich möglichst kontinuierlich an veränderliche Strömungen angepasst werden können.
Da diese Eigenschaften von Rechengittern aus Advancing-Front- oder Delaunay-Triangulationen nicht besonders gut erfüllt werden, werden zwei andere Konzepte, basierend auf einem Quadtree und auf Voronoi-Diagrammen, entworfen und getestet. Es wird gezeigt, dass Voronoi-Diagramme Eigenschaften besitzen, die sie im Kontext von DSMC anderen Gittertypen überlegen erscheinen lassen.
Die Teilchenverfolgung hat aufgrund der großen Teilchenzahl einen wesentlichen Anteil an der Rechenzeit der Simulation. Da die Möglichkeiten zur Verfolgung der Teilchen von der Art des Rechengitters abhängen, werden für beide Konzepte effiziente Teilchenverfolgungsalgorithmen vorgeschlagen.

Stichwörter: DSMC, Teilchenverfolgung, Rechengitter, Voronoi-Diagramm, Delaunay-Triangulation, kd-Baum, Quadtree

Abstract

The DLR is currently developing a DSMC software to simulate dilute gas flows in three-dimensional geometries. The gas is understood as a large number of particles moving through the computational grid, hitting faces and colliding with each other. This task requires a computational grid consisting of convex and near-spherical cells. The density of the grid should be adjustable to changing flow conditions both locally and in time as continuous as possible.
Since these characteristics are not well satisfied by advancing front or Delaunay triangulations, two other approaches, one based on a quadtree and one based on Voronoi diagrams, will be designed and tested. It is shown that as a result of their properties Voronoi diagrams seem to be very well suited for the DSMC-method. Due to the large number of particles the particle tracking causes a substantial part of the computing time. As the possibilities of tracking depend on the type of the grid, efficient particle tracking algorithms will be proposed for both concepts.

Keywords: DSMC, particle tracking, computational grid, Voronoi diagram, Delaunay triangulation, kd-tree, quadtree

Inhaltsverzeichnis

1 Einleitung

Mit dem DSMC-Verfahren lassen sich Strömungen bei sehr niedriger Dichte simulieren, wie sie z.B. bei einem Flug durch die obere Atmosphäre oder bei Expansionsströmungen aus Lageregelungstriebwerken von Satelliten auftreten. Bei diesen verdünnten Gasströmungen kann nicht mehr von einem Kontinuum ausgegangen werden. Deshalb wird die Strömung als eine sehr große Menge einzelner Partikel aufgefasst, deren Bewegungen sowie Stöße untereinander und Kollisionen mit einer Randkontur simuliert werden. Das Verfahren selbst und die begrenzte Rechenleistung erfordern eine Diskretisierung des simulierten Raumes durch ein Rechengitter. Sowohl die Qualität der Ergebnisse als auch die zu ihrer Berechnung benötigte Zeit kann durch die Qualität des Rechengitters erheblich beeinflusst werden.
Einige dreidimensionale Problem können durch Ausnutzung von Symmetrien in ein oder zwei Dimensionen wesentlich leichter behandelt werden. Am DLR wird derzeit ein DSMC-Programm entwickelt, das in drei Dimensionen arbeiten kann, damit auch solche Probleme behandelt werden können, die keine geeignete Symmetrie zur Vereinfachung besitzen. In drei Dimensionen wird einerseits die Diskretisierung des Raumes komplizierter, andererseits werden schlechte Eigenschaften des Rechengitters einen größeren negativen Einfluss ausüben. Deshalb wird ein Konzept aus Rechengitter inklusive Teilchenverfolgung benötigt, das auch in drei Dimensionen einfach und effizient ist und die spezifischen Anforderungen des DSMC-Verfahrens so gut wie möglich erfüllt.
In Kapitel 2 wird die Funktionsweise des DSMC-Verfahrens beschrieben um damit die Notwendigkeit eines Rechengitters zu begründen. Aus der Notwendigkeit wird in Kapitel 3 auf Eigenschaften geschlossen, die ein für DSMC ideales Rechengitter erfüllen müsste und verschiedene Gittertypen und einige Methoden zur Generierung von Rechengittern vorgestellt. Auf dieser Grundlage werden im Kapitel 4 zwei Ideen entworfen, welche dann in den Kapiteln 5 und 6 zu zwei vollständigen Rechengitterkonzepten zur Teilchenverfolgung in DSMC ausgebaut werden. Beide Konzepte wurden für zwei Dimensionen in Form eines einfachen experimentellen Programms

realisiert und getestet. In Kapitel 7 wird deren Implementierung beschrieben und deren Effizienz durch Laufzeitmessungen abgeschätzt. Den Schluss bildet eine Bewertung der Konzepte und ein Ausblick auf mögliche Weiterentwicklungen des aussichtsreichsten Konzeptes.
Der Einfachheit halber erfolgen alle Beschreibungen so weit dies möglich ist in zwei Dimensionen. Häufig muss für eine Erweiterung auf drei Dimensionen nur „Kanten“ durch „Flächen“ ersetzt werden. Es gibt aber eine Reihe von geometrischen Phänomenen, die solche trivialen Erweiterungen von zwei auf drei Dimensionen verhindern. Da die Konzepte für drei Dimensionen geeignet sein sollen, wird auf Unterschiede zwischen zwei und drei Dimensionen hingewiesen und auf Lösungen, die nur in zwei Dimensionen funktionieren würden, verzichtet.

2 Das DSMC-Verfahren

2.1 Das Modell

Zur physikalischen Beschreibung von Strömungen gibt es zwei grundsätzlich verschiedene Ansätze. Entweder man betrachtet das strömende Gas als ein Kontinuum, dann lassen sich die interessanten Größen wie Geschwindigkeit, Dichte und Temperatur als Lösungen der Navier-Stokes-Gleichungen gewinnen. Dabei muss allerdings annähernd thermodynamisches Gleichgewicht vorausgesetzt werden. Oder man betrachtet das Gas im Sinne der kinetischen Gastheorie als eine große Anzahl miteinander wechselwirkender Teilchen und gewinnt die interessanten Größen aus der statistischen Verteilung der Zustände der einzelnen Teilchen.
Die kinetische Gastheorie ist der allgemeinere Ansatz, wird aber aufgrund ihrer Komplexität nur eingesetzt, wenn die Voraussetzungen für die Navier-Stokes-Gleichungen nicht mehr erfüllt sind. Eine Entscheidung lässt sich z.B. mit Hilfe der Knudsen-Zahl treffen, welche als Verhältnis der mittleren freien Weglänge zu einer für die Strömung charakteristischen Länge definiert ist. Bei sehr kleinen Knudsen-Zahlen ($\mathrm{Kn} < 0,001$) kann man von thermodynamischem Gleichgewicht ausgehen. Da die charakteristische Länge aber wie bei vielen dimensionslosen Größen unterschiedlich gewählt werden kann, ist eine eindeutige Abgrenzung schwierig. Einen Überblick verschiedener Verfahren und ihre Bewertung findet man bei Macrossan [14].
Die DSMC-Methode (Direct-Simulation-Monte-Carlo) basiert auf der Sichtweise der kinetischen Gastheorie, umgeht aber eine direkte Lösung der Boltzmann-Gleichung, indem die Bewegungen und Kollisionen der einzelnen Teilchen innerhalb eines „virtuellen Experiments“ berechnet und statistisch analysiert werden.
Da die exakte Berechnung aller Moleküle aufgrund der hohen Anzahl und insbesondere wegen der Stoßprozesse untereinander zu aufwändig wäre, wird das Modell üblicherweise vereinfacht:

- Um die zu simulierende Teilchenzahl zu reduzieren, wird eine größere Anzahl Gas-Moleküle zu jeweils einem Simulationspartikel zusammengefasst.
- Bewegungen und Stoßprozesse werden unabhängig voneinander betrachtet. Nach einer stoßfreien Bewegung innerhalb eines Zeitschritts Δt folgt die stochastische Behandlung der Stöße auf Basis des aktuellen Zustands der Teilchen.
- Zur Bestimmung der Teilchen, die einen Stoß erfahren, werden nicht alle möglichen Stoßpartner überprüft, sondern zu jedem Simulationsteilchen die Stoßwahrscheinlichkeit mit nur einem anderen Teilchen aus der näheren Umgebung bestimmt. Die Stoßwahrscheinlichkeit kann als Anteil der stoßenden Gas-Teilchen aus der Menge der durch die beiden Simulationsteilchen repräsentierten Teilchen verstanden werden.
- Auf Stöße mit mehr als zwei beteiligten Teilchen wird aufgrund der geringen Wahrscheinlichkeit zunächst verzichtet. Nur wenn auch chemische Reaktionen mit mehr als zwei Komponenten berücksichtigt werden sollen, müssen auch Stöße mehrerer Teilchen zugelassen werden.

2.2 Notwendigkeit zur Diskretisierung des Raumes

Obwohl das oben beschriebene Modell anschaulich sehr einfach ist, ergeben sich bei der Realisierung einige Schwierigkeiten.
Unter den genannten Vereinfachungen würde für einzelne Teilchen keine realitätsnahe Simulation entstehen. Ihre Berechtigung erhalten sie erst dadurch, dass man sich im Ergebnis nicht für die Bewegung einzelner Teilchen, sondern nur für die Statistik sehr vieler Teilchen über eine große Anzahl von Iterationen interessiert.
Um das „statistische Rauschen“ zu verringern, muss die Anzahl der tatsächlich simulierten Teilchen groß sein, obwohl auch durch wenige simulierte Partikel beliebig viele Gas-Moleküle repräsentiert werden könnten. Allerdings wird auch der Rechenaufwand mit wachsender Teilchenzahl sehr groß. Dies hat im wesentlichen drei Gründe:
Die Bewegung: Bei der Bewegung jedes Teilchens muss nicht nur seine Position aktualisiert werden, sondern auch mögliche Kollisionen mit den Kanten der Randkontur des Simulationsgebietes erkannt werden. Die Randkontur umfasst dabei nicht nur die Außengrenzen, sondern auch innere Ränder, welche die umströmten Körper darstellen.

Durch eine Zerlegung des Simulationsgebietes in Zellen lässt sich die Anzahl der auf Kollision zu prüfenden Kanten reduzieren. Dabei entstehen zusätzliche, für die Teilchen transparente Kanten, die ebenfalls (auf Durchdringung) überprüft werden müssen. Die Anzahl der bei der Bewegung eines Teilchens zu überprüfenden Kanten hängt aber nur noch von der Randkontur der Zelle und nicht mehr von der Randkontur des ganzen Simulationsgebietes ab, welche bei komplizierten Körpern schnell aus einer großen Menge geometrischer Primitive bestehen kann.

Der Stoßprozess: Nach jedem Bewegungsschritt muss zur Behandlung der Stoßprozesse für jedes Teilchen bekannt sein, welche anderen Teilchen sich in der näheren Umgebung befinden.

Eine einfache Idee dazu wäre z.B. die Bestimmung der nächsten Nachbarn, wobei mit der Teilchenzahl N die Rechenzeit mindestens mit $N \log N$ anwächst. Ein Beweis für dieses bekannte Problem ist z.B. bei R. Klein [11] zu finden.

Durch Zerlegung des Raumes in Gebiete lässt sich hier nicht nur die miteinander zu vergleichende Teilchenzahl verringern, die Überprüfung auf Nähe lässt sich sogar ganz vermeiden, da alle Teilchen in dem Gebiet, unter bestimmten Voraussetzungen an Form und Größe des Gebietes, automatisch eine gewisse Nähe aufweisen.

Die Auswertung: Nach einer Bewegung aller Teilchen muss ihr Zustand noch statistisch analysiert werden, um z.B. Geschwindigkeit, Druck und Temperatur zu bestimmen. Für die örtliche Auflösung wird wiederum eine lokal zusammengehörige Menge von Teilchen benötigt. Würde man dazu einfach um einen Punkt eine Kreis ziehen und prüfen, welche Teilchen sich innerhalb dieses Kreises befinden, würde die Rechenzeit wieder mehr als linear von der Anzahl der Teilchen abhängen, was sich ebenfalls durch eine Einteilung in Gebiete vermeiden lässt.

Bei naiven Methoden würde der Rechenaufwand für Bewegung und Stoß jedes einzelnen Teilchens und für die Auswertung nach jedem Zeitschritt stark von der Gesamtanzahl der Teilchen und der Komplexität der Körper abhängen. Diese Abhängigkeit kann durch eine geschickte Diskretisierung des Raumes deutlich verringert und dadurch Strömungen in realistischen Geometrien effizient simuliert werden.

3 Rechengitter

3.1 Ein ideales Gitter für DSMC

Aus den drei in Kapitel 2.2 beschriebenen Aufgaben ergeben sich zunächst unterschiedliche Anforderungen an die räumliche Zerlegung.
Da zu jeder Zelle bekannt sein muss, welche Teilchen sie enthält, braucht man entweder eine eindeutige Abbildung von den Koordinaten eines Teilchens zu der Zelle, oder man verfolgt die Bewegung des Teilchens von einer Zelle zur nächsten. Eindeutige Abbildungen gibt es nur in regulären Gittern (vgl. Kap. 3.2). Soll das Teilchen durch die Zellen verfolgt werden, sollte das Gitter ausschließlich aus konvexen Zellen mit geradlinigen Kanten bestehen. Nicht-geradlinige Kanten führen automatisch zu nicht-konvexen Zellen, wodurch die Teilchenverfolgung (Kap. 6.2) deutlich komplizierter wird.
Die Kollisionen mit der Randkontur können entweder dadurch realisiert werden, dass nur der Teil der Randkontur überprüft wird, welcher sich innerhalb der Zelle befindet, oder die Randkontur wird mit undurchlässigen Zellkanten abgebildet. Dazu müssten die Zellen aber an beliebige Randkonturen angepasst werden können.
Des weiteren wäre es von Vorteil, wenn die Zellen keine indirekten Nachbarn hätten. Zum Beispiel hat eine Zelle in einem Rechteckgitter vier direkte Nachbarn, mit denen sie eine gemeinsame Kante hat, und vier indirekte Nachbarn, mit denen sie nur einen gemeinsamen Punkt hat. Ein Teilchen, das sich an der Ecke des Rechtecks knapp vorbei bewegt, müsste auch für einen sehr kurzen Weg die komplette Zelle des indirekten Nachbarn durchqueren. In einem 3D-Gitter aus Quadern hätte jede Zelle bereits sechs direkte Nachbarn, 12 indirekte Nachbarn mit einer gemeinsamen Kante und weitere acht indirekte Nachbarn mit einem gemeinsamen Punkt.
Die Behandlung des Stoßprozesses und die statistische Auswertung stellen Anforderungen an Form und Größe der Zelle. Für die Auswertung ist die maximal erreichbare örtliche Auflösung der Ergebnisse durch die Größe der Zellen bestimmt. Für den Stoßprozess sollten die Zellen einerseits nicht deutlich kleiner als die mittlere freie

Weglänge sein, damit die Teilchen nicht in jedem Zeitschritt unnötig viele Zellen durchqueren müssen. Andererseits sollte die nominelle Größe der Zelle auch nicht wesentlich größer sein als die mittlere freie Weglänge, damit potentielle Stoßpartner nicht außerhalb der Reichweite des jeweiligen Teilchens liegen. Dazu sollte die Form der Zelle so beschaffen sein, dass sie nicht in verschiedenen Richtungen völlig unterschiedliche Ausdehnungen hat. Außerdem sollten die Zellen dazu konvex sein. Wenn eine Zelle nicht-konvex ist, weil sie beispielsweise an einer spitzen Ecke des Randes liegt, könnten Teilchen von beiden Seiten dieser Ecke miteinander stoßen, obwohl sie durch eine Wand voneinander getrennt sind.
Da die optimale Form und Größe der Zellen also vom aktuellen Zustand der Strömung und damit von Zeit und Ort abhängig ist, sollte das Gitter mit möglichst wenig Aufwand nach bestimmten Kriterien an veränderte Bedingungen angepasst werden können.

Ein für DSMC ideales Gitter besteht also aus konvexen Zellen ohne indirekte Nachbarn. Die Ausdehnung der Zellen ist möglichst isotrop und entspricht der mittleren freien Weglänge, auch wenn sich diese lokal und zeitlich verändert. Da sich diese Anforderungen auf den ersten Blick nicht grundsätzlich widersprechen, ist es nahe liegend, aber nicht unbedingt notwendig, den Raum durch nur ein Gitter einzuteilen und dieses auf alle drei Probleme (Bewegung, Stoß und Auswertung) gleichzeitig anzuwenden. Bei mehreren Gittern wird immer zusätzlicher Aufwand entstehen, weil diese konstruiert und gespeichert werden müssen und weil nach einem Bewegungsschritt für jedes Teilchen in jedem der Gitter bekannt sein muss, in welcher Zelle sich das Teilchen gerade befindet. Ideal für DSMC wäre also ein einzelnes Gitter, das möglichst viele dieser Anforderungen möglichst gut erfüllt.

3.2 Klassifikation

Üblicherweise unterscheidet man **strukturierte** und **unstrukturierte Gitter**. Der Unterschied besteht darin, dass strukturierte Gitter eine regelmäßige Topologie besitzen. Praktisch bedeutet das, dass für jede Zelle berechnet werden kann, mit welchen Zellen sie benachbart ist.
Die strukturierten Gitter lassen sich weiter in **reguläre** und **irreguläre Gitter** einteilen.
Ein reguläres Gitter zeichnet sich dadurch aus, dass alle Zellen die gleiche Geome-

trie besitzen. Ein irreguläres Gitter kann Zellen unterschiedlicher Form haben, es existiert aber immer eine Abbildung auf ein reguläres Gitter.
Unstrukturierte Gitter sind der allgemeinste Gittertyp und daher nicht immer näher zu klassifizieren. Wenn das Gitter aus nur einer Sorte von Polygonen besteht, wird es nach diesen benannt. Das wichtigste Beispiel dafür sind Triangulationen für Gitter, die nur aus Dreiecken bestehen. „Tetrahedrisationen“ werden meist auch Triangulation genannt.
Im Allgemeinen gilt, dass ein Gitter mit zunehmenden Einschränkungen einfacher, aber auch unflexibler wird. Um hier einen Kompromiss zu finden, bieten sich als dritte Klasse **Kombinationen** verschiedener Gittertypen an. Diese sind meist hierarchisch aufgebaut, so dass in jeder Zelle wieder ein untergeordnetes Gitter existieren kann. Denkbar sind aber auch viele andere Kombinationsmöglichkeiten wie überlappende Gitter, ineinander übergehende Gitter oder unstrukturierte Gitter, die nach einem Hintergrundgitter aufgebaut sind.

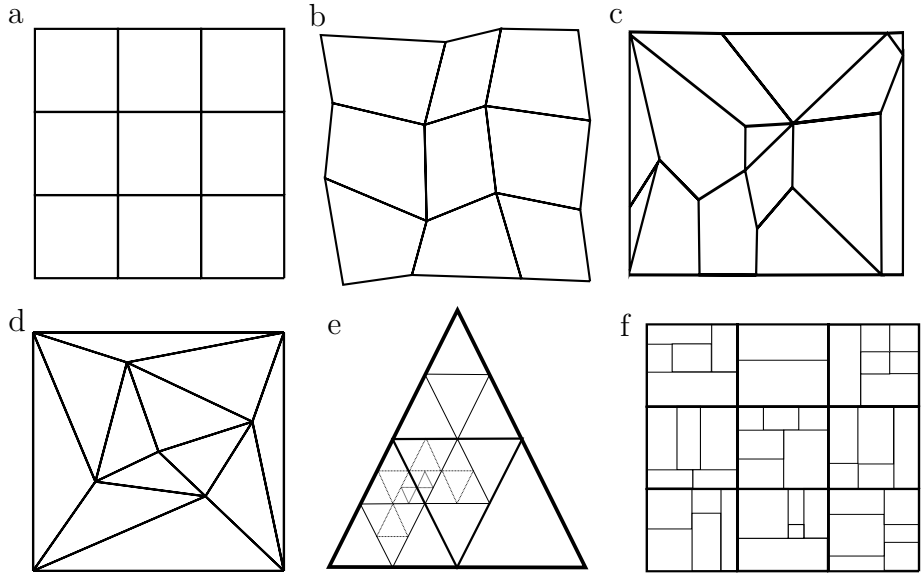

Abb. 3.1: Beispiele: a) regulär; b) irregulär; c) unstrukturiert; d) unstrukturierte Triangulation; e) hierarchisches Gitter aus strukturierten Triangulationen; f) hierarchisches Gitter aus einem regulären und mehreren irregulären Rechteckgittern

3.3 Geometrie

3.3.1 Reguläre Gitter

Die Vorteile strukturierter regulärer Gitter sind, dass die Vorgabe sehr weniger Informationen (z.B. die Angabe einer Kantenlänge) ausreicht, um das Gitter eindeutig zu definieren und dass dadurch die geometrischen Eigenschaften jeder Zelle und die Beziehungen unter den Zellen bekannt sind.
Die wenigsten Informationen werden für Gitter aus regelmäßigen Polygonen benötigt. Um eine Ebene damit lückenlos auszufüllen, gibt es genau drei Möglichkeiten. Dies sind regelmäßige Dreiecke, Vierecke und Sechsecke. Die Liste der raumfüllenden Polyeder ist etwas länger, aber auch hier genügt die Angabe einer Größe, wie z.B. der Kantenlänge, um das ganze Gitter zu beschreiben.
Das Dreieck hat unter den regulären Gittern den spitzesten Winkel und das kleinste Verhältnis von Flächeninhalt zu Umfang. Optimal in diesen beiden Eigenschaften ist das regelmäßige Sechseck, weshalb es vergleichsweise oft in der Natur, z.B. als Benzolringe oder Bienenwaben, anzutreffen ist. Ein Beweis, dass es in dieser Gruppe kein Gitter gibt, welches das Fläche-Umfang-Verhältnis des Sechseckgitters übertrifft, gelang erst 1999 durch Thomas Hales [6]. Außerdem gibt in einem Gitter aus regelmäßigen Sechsecken im Gegensatz zu Drei- und Vierecken keine indirekten Nachbarn.
Keines der regulären Gitter lässt sich aber an beliebigen Randkonturen oder an lokale Strömungsbedingungen anpassen (abgesehen von Methoden wie konformen Abbildungen, die aber immer zu nicht-geraden Kanten und damit nicht-konvexen Zellen führen). Würde man z.B. mit einen Rechteckgitter einen Kreis wie eine Stufenfunktion abbilden, würde sich die Richtungsänderung von Teilchen bei Kollisionen mit dieser Stufenfunktion auch bei beliebig feiner Auflösung nicht dem Verhalten an einem Kreis annähern.
Aufgrund seiner Einfachheit eignet sich ein Rechteckgitter allerdings zu Kombinationen mit anderen Gittern, und die reguläre Sechseckstruktur könnte als Vorbild für ein unstrukturiertes Gitter mit ähnlich guten Eigenschaften dienen.

3.3.2 Irreguläre Gitter

Im Unterschied zu den regulären Gittern hängt die Menge der zur Definition des Gitters benötigten Informationen (in der Ebene quadratisch, im Raum kubisch) von der Anzahl der Zellen ab und kann z.B. aus der Angabe der Koordinaten jedes Kno-

tenpunktes bestehen.
Durch die Verteilung der Knotenpunkte können mit den Kanten der Zellen zwar beliebige Formen abgebildet werden, aber nicht mit beliebiger Auflösung. Ebenso kann die Größe der Zellen variabel an lokale Strömungsbedingungen angepasst werden, aber eine Verfeinerung des Gitters in einem Bereich führt immer zu einer Vergröberung in einem anderen Bereich, weil die Gesamtanzahl der Zellen nicht verändert werden kann. Dieses Problem könnte man umgehen, indem man Zellen ohne Ausdehnung zulässt. Dadurch könnte man beliebig einzelne Zellen quasi hinzufügen oder entfernen. Ein solches Gitter wäre fast so flexibel wie ein unstrukturiertes Gitter ohne dass es eine komplizierte Topologieverwaltung benötigt. Dies könnte aber unter Umständen sehr ineffizient sein, weil es viele nutzlose Zellen besitzen kann.

3.3.3 Unstrukturierte Gitter

Ein unstrukturiertes Gitter besitzt keine Eigenschaft, die es verhindert, jede Randkontur beliebig genau abzubilden und jederzeit an den Strömungszustand angepasst zu werden. Die Schwierigkeit besteht allerdings darin, ein solches Gitter zu erzeugen und zu verändern sowie die durchschnittliche und die schlechteste auftretende Form von Zellen vorher abzuschätzen.
Eine sehr einfache Methode, ein unstrukturiertes Gitter zu erzeugen, ist die Zerlegung eines Polygons mit n Ecken in eine Triangulation mit n Knoten, indem so lange wie möglich Ecken über Diagonalen verbunden werden, ohne dass sich die Diagonalen schneiden. Dabei können aber sehr spitzwinklige Dreiecke entstehen. Außerdem lässt sich die Methode nicht ohne weiteres auf den dreidimensionalen Raum erweitern, da sich nicht jedes Polyeder in Tetraeder zerlegen lässt, ohne zu den Ecken weitere Knoten (so genannte Steiner-Punkte) hinzuzufügen. Ein einfaches Beispiel ist das Schönhardt-Polyeder [17].
Mit einer gegebenen Triangulation eines Polygons lässt sich nach einer einfachen Methode von Hertel und Mehlhorn [8] eine Zerlegung in wenige konvexe Teile finden. Dabei wird so oft wie möglich eine Diagonale entfernt, ohne dass dadurch ein nicht-konvexes Polygon entsteht. Diese Zerlegung ist in vielen Fällen besser als die Triangulation, kann aber auch extreme Formen annehmen. Wenn das Polygon zu großen Teilen nicht-konvex ist, wird die Triangulation gar nicht erst verändert und wenn ein vollständig konvexes Polygon zerlegt wurde, erhält man mit dieser Methode nur eine Zelle, die dem Ausgangspolygon entspricht. Im folgenden Kapitel werden deshalb drei geeignetere Methoden zur Erzeugung unstrukturierter Gitter vorgestellt.

3.4 Methoden zur Gittergenerierung

3.4.1 Advancing-Front-Triangulation

Die Advancing-Front-Methode erzeugt ebenfalls eine Triangulation eines Gebietes, das durch eines oder mehrere Polygone vorgegeben wird. Dabei wächst das Gitter mit der so genannten Advancing-Front vom Rand des Gebietes Richtung Mitte.

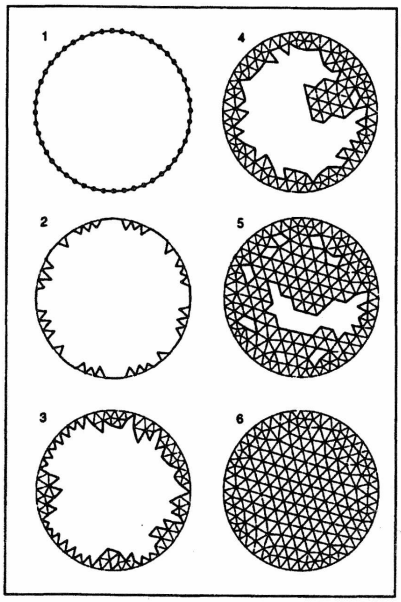

Abb. 3.2:
Der Advancing-Front-Prozess in mehreren Stufen (aus [18])

Es gibt viele Varianten der Advancing-Front-Methode, die sich in den Voraussetzungen und dem Verhalten unterscheiden. Üblicherweise muss eine Punkteverteilung entlang der Randkontur vorgegeben werden, welche die „Advancing Front“ in ihrem ersten Schritt darstellt. Daraus wird eine Kante, z.B. die kürzeste, als Basis für ein neues Dreieck ausgewählt. Je nach Winkel, den diese Kante mit ihren beiden Nachbarkanten bildet, wird entweder die Kante mit einer Nachbarkante zu einem Dreieck verbunden oder ein neuer Knotenpunkt vor der Advancing-Front erzeugt, der mit der Basiskante das neue Dreieck bildet. Die jetzt innerhalb des Netzes liegenden Kanten werden aus der Advancing-Front entfernt und die neuen Kanten hinzugefügt. Dieser Vorgang wird solange wiederholt, bis die Advancing-Front nur noch aus drei Kanten besteht.

Die Form des Gitters kann durch verschiedene Verfahren beeinflusst werden, von denen Peraire in [18] einige beschreibt. Peraire lässt das Gitter zunächst in einem normalisierten Raum erzeugen, in dem die Punkte der Advancing-Front in etwa den selben Abstand zueinander haben, und transformiert das Gitter dann zurück in den ursprünglichen Raum. Die Dichte des Gitters lässt sich dabei durch ein vorgegebenes Hintergrundgitter steuern. Im fertigen Gitter können dann noch die Positionen der Knotenpunkte z.B. durch eine Laplace-Glättung verändert werden oder einzelne Dreiecke durch Umklappen von Kanten besser verbunden werden.

3.4.2 kd-Baum

Ein kd-Baum ist zunächst einmal eine Datenstruktur, die in sehr unterschiedlichen Bereichen angewendet wird. Die Idee des kd-Baums ist, eine gegebene Menge in Teile zu zerlegen und jedes einzelne der Teile weiter zu zerlegen. In der dabei entstehenden Hierarchie wird die initiale Menge Wurzel genannt, die Untermengen sind innere Knoten (Äste) und die Knoten, von denen keine weiteren Äste ausgehen, werden Blätter genannt.
In geometrischen Anwendungen werden als initiale Menge meist Rechtecke benutzt, die jeweils in zwei (Binärbaum) oder vier (Quadtree) weitere Rechtecke eingeteilt werden. Möglich sind aber auch andere Formen wie z.B. Dreiecke (vgl. Abb. 3.1 e). Jede Zelle kann ein weiteres Gitter enthalten, sodass eine Hierarchie aus regulären Gittern entsteht.
Wie anfangs bereits erwähnt, reichen reguläre Gitter allein nicht aus, um beliebige Ränder abzubilden. Daher müssen zumindest die Blätter, die einen Teil des Randes enthalten, gegenüber den anderen Blättern gesondert betrachtet werden. Häufig wird der Baum dazu so weit eingeteilt, dass jedes Blatt maximal eine Ecke enthält, weil daraus leicht eine Triangulation erzeugt werden kann, indem die Ecken dieser Zellen mit der Ecke des Randes verbunden werden und die übrigen Zellen durch ihre Diagonale geteilt werden.

3.4.3 Delaunay-Triangulation

Das Idee der Delaunay-Triangulation ist, eine gegebene Punktmenge so zu Dreiecken zu verbinden, dass dabei die minimalen Winkel der Dreiecke maximiert werden. Zur Konstruktion der Delaunay-Triangulation lässt sich verwenden, dass die kleinsten Winkel genau dann maximal sind, wenn der Umkreis der Punkte eines jeden Dreiecks keinen anderen Punkt enthält (Beweis siehe z.B. [11]). Diese Bedingung wird Delaunay-Kriterium oder Umkreisbedingung genannt.

Voronoi-Diagramm

Mathematisch beschrieben wurden Voronoi-Diagramme zuerst von Dirichlet und dann in n-dimensionaler Verallgemeinerung durch Voronoi. Die grundlegende Idee ist allerdings schon wesentlich älter. Eine Menge wird so in Teilmengen zerlegt, dass innerhalb jeder Teilmenge der Einfluss ihres Zentrums größer ist als der Einfluss

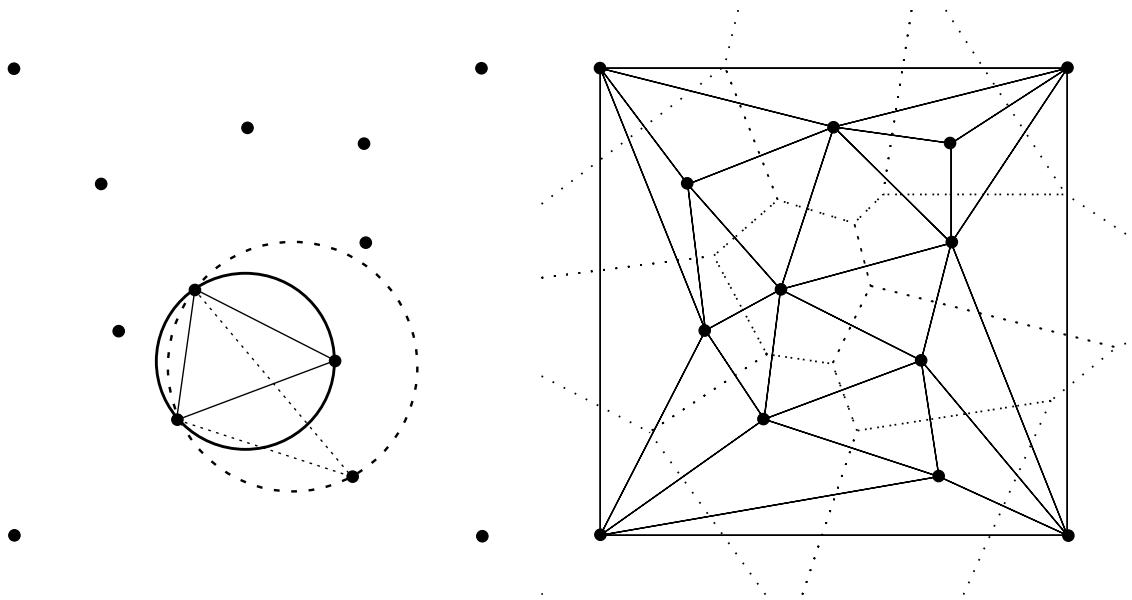

Abb. 3.3: Links: Konstruktion der Delaunay-Triangulation mit Umkreiskriterium
Rechts: Fertige Delaunay-Triangulation und entsprechendes Voronoi-Diagramm

aller anderen Zentren.
Aufgrund dieser sehr allgemeinen Definition lassen sich viele Strukturen in der Natur als Voronoi-Diagramm interpretieren. Beispiele dafür sind das Muster auf dem Fell einer Giraffe, ein Spinnennetz oder in 3D die Form der Bläschen in Schaum. Außerdem wird es in den verschiedensten Bereichen eingesetzt, unter anderem den Naturwissenschaften (bspw. Brillouin-Zonen) oder in der Wirtschafts- bzw. Geschichtswissenschaft und der Städteplanung für die Einflussbereiche von Märkten, Siedlungsgebieten oder Feuerwehren. Einen breiten Überblick über viele Varianten und Anwendungsgebiete von Voronoi-Diagrammen gibt A. Okabe [16].
In der Geometrie ist die zu zerlegende Menge üblicherweise die Ebene bzw. der Raum und die Zentren sind oft durch Punkte gegeben (möglich sind aber auch Linien und andere Objekte). Der Einfluss ist häufig durch die euklidische Metrik definiert. Weitere Möglichkeiten sind z.B. auch die anderen Minkowsky-Metriken

$$L_p : d(\vec{x}, \vec{y}) = \left[\sum_{i=1}^{n} |x_i - y_i|^p \right]^{\frac{1}{p}} \quad . \tag{3.1}$$

Die Linie, auf die zwei Zentren den gleichen Einfluss ausüben, wird der Bisektor der beiden Zentren genannt. Bildet man für ein Zentrum wie in Abbildung 3.4 die Bisektoren zu allen anderen Zentren und schneidet die Ebene hinter diesen Bisektoren ab, bleibt die Voronoi-Zelle des Zentrums übrig. Die Menge aller Voronoi-Polygone, in anderen Worten die Menge aller Punkte, auf die zwei oder mehr Zentren den gleichen Einfluss ausüben, bildet eine lückenlose Zerlegung der Ebene, das Voronoi-Diagramm.

Dualität

Unter der Voraussetzung, dass nicht mehr als drei Punkte auf einem Kreis liegen, sind die Delaunay-Triangulation und das Voronoi-Diagramm dual zueinander. Wenn die eine Darstellung bekannt ist, kann die andere daraus abgelesen werden.
Verbindet man im Voronoi-Diagramm diejenigen Zentren miteinander, deren Zellen eine gemeinsame Kante haben, erhält man die Delaunay-Triangulation. Verbindet man in der Delaunay-Triangulation zu jedem Zentrum die Schnittpunkte der Mittelsenkrechten der Delaunay-Kanten oder die Mittelpunkte der Umkreise aller Dreiecke miteinander, erhält man das Voronoi-Diagramm. Wenn vier oder mehr Punkte auf einem Kreis liegen, muss zwischen Delaunay-Gitter und Delaunay-Triangulation unterschieden werden. Das Delaunay-Gitter ist dual zum Voronoi-Diagramm und kann neben Dreiecken auch andere Polygone enthalten. Durch die Nachtriangulation dieser stets konvexen Polygone entsteht die Delaunay-Triangulation.

3.4.4 Kombinationen

Diese Methoden werden häufig kombiniert, um die Vorteile verschiedener Techniken zu nutzen.
In größeren Delaunay-Triangulationen kann die Lokalisierung eines Punktes im Gitter viel Zeit in Anspruch nehmen. Wenn man die Zentren in einen kd-Baum einsortiert, lässt sich die Lokalisierung deutlich effizienter gestalten („bucketing“-Technik [16]).
Um die Delaunay-Triangulation eines berandeten Gebietes zu erzeugen, können die dazu benötigten Punkte aus Advancing-Front-Verfahren gewonnen werden. Dadurch erhält man eine an den Rand angepasste Punkteverteilung und gleichzeitig die mathematischen Eigenschaften der Delaunay-Triangulation.

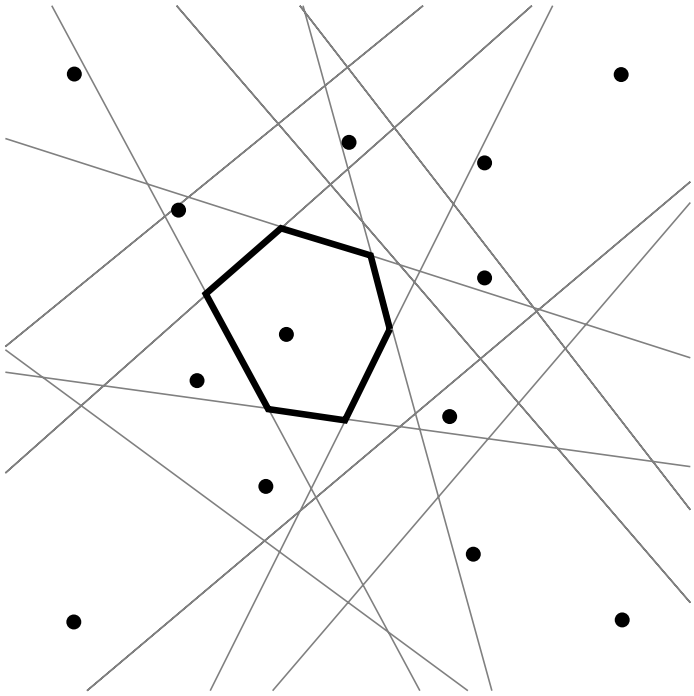

Abb. 3.4: Konstruktion eines Voronoi-Polygons

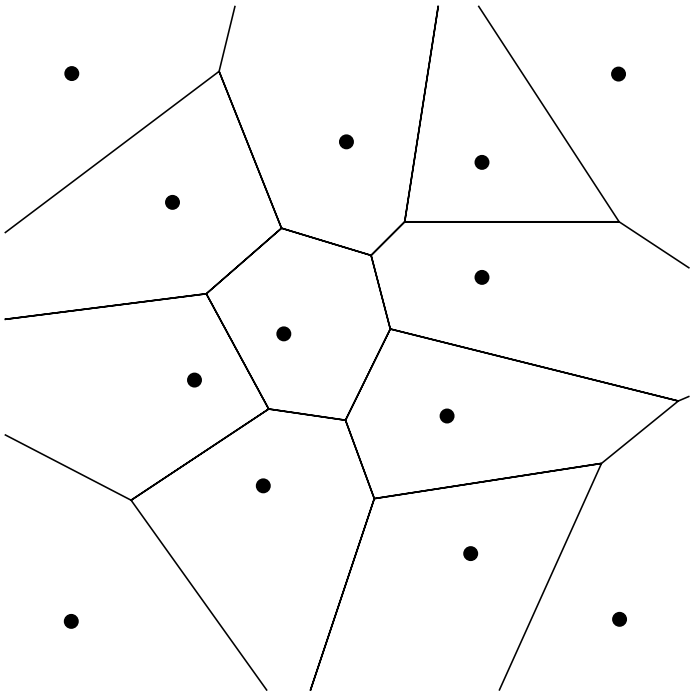

Abb. 3.5: Fertiges Voronoi-Diagramm

4 Auswahl der Methoden

4.1 Advancing-Front-Methode

Da die Advancing-Front-Methode das Gitter vom Rand nach innen aufbaut, können mit dieser Methode sehr komplizierte Geometrien vernetzt werden. Da die Knotenpunkte des Gitters erst während des Aufbaus erzeugt werden, kann die Entstehung von Dreiecken mit sehr spitzen oder stumpfen Winkeln weitestgehend verhindert werden.
Als eine Triangulation hat die Advancing-Front-Methode den weiteren Vorteil, dass einige Berechnungen auf einem Dreieck, dem einfachsten aller Polygone, besonders einfach sind. Es gibt z.B. einen Teilchenverfolgungsalgorithmus von Löhner [13], der in den natürlichen Koordinaten des Dreiecks bestimmt, ob ein Teilchen während der Bewegung das Dreieck verlässt.
Aus diesen Gründen wird die Advancing-Front-Methode häufig zur Gittererstellung für Finite-Elemente- und Finite-Volumen-Verfahren eingesetzt und ist weit verbreitet in der Vernetzung von Oberflächen in CAD-Anwendungen.
Leider lassen sich mit diesem Verfahren nur Dreiecke bzw. Tetraeder erzeugen. Nach einem ähnlichen Prinzip lassen sich zwar auch Viereck- statt Dreieckgitter erzeugen („paving“-Technik [3, 10]), dies sind aber ganz eigenständige Methoden und keine natürlichen Erweiterungen der Advancing-Front-Triangulation.
Ein anderer Nachteil ist, dass das Gitter zwar nach einer vorgegebenen Dichte erzeugt werden kann, nachträgliches Vergröbern oder Verfeinern eines bereits erzeugten Gitters aber zusätzliche Strategien erfordert. Eine teilweise Neuvernetzung ist zwar möglich, aber für DSMC nicht sinnvoll, weil dabei jedes Teilchen in dem Gebiet neu eingeordnet werden müsste. Eine bessere Möglichkeit wäre das Verschieben von Knoten oder Löschen und Hinzufügen von neuen Dreiecken, wobei aber die Gefahr besteht, dass die Qualität des Netzes darunter leidet.
Eine Advancing-Front-Triangulation ist also einfach, robust und wird bereits für DSMC-Simulationen eingesetzt [12]. Da Dreiecke dem für DSMC idealen Gitter

nicht sehr nahe kommen und bei Adaptionen des Gitters nur schwer sichergestellt werden kann, dass die für Dreiecke zunächst gute Qualität der Advancing-Front-Triangulation erhalten bleibt, wird sie hier nicht weiter verfolgt.

4.2 kd-Baum

Das Interessante an einem Gitter aus dieser Datenstruktur ist die tiefe Hierarchie, die für die Teilchenverfolgung einen Vor- und einen Nachteil hat.
Der entscheidende Nachteil ist, dass im kd-Baum keine vollständige Topologie zur Verfügung steht. Zu einer bestimmten Zelle sind erstmal nur die Nachbarzellen bekannt, die zusammen mit dieser Zelle aus einer gemeinsamen Vorgängerzelle hervorgegangen sind. Im schlimmsten Fall ist die gemeinsame Vorgängerzelle aller Nachbarzellen die Wurzel des kd-Baums.
Es ist außerdem nicht möglich, die fehlende Topologie zu ergänzen und ab dann nur noch die Blätter des kd-Baums als Gitter zu verwenden, da zu einer Kante einer Zelle mehrere Nachbarzellen gehören können (vgl. Abb. 5.1). Immerhin relativiert die fehlende Topologie das Problem der indirekten Nachbarn und man kann ohne große Nachteile die einfachste Form aus Rechtecken benutzen.
Der Vorteil ist, dass man aus den Koordinaten sehr einfach auf die entsprechende Zelle schließen kann, indem man die Position von der Wurzel bis zum Blatt verfolgt. Dadurch lassen sich ohne viel Mehraufwand verschiedene Netzdichten für die Teilchenverfolgung und für die Auswertung, je nach ihren speziellen Anforderungen, verwenden.
Da beide Eigenschaften nicht von der Anzahl der Unterteilungen abhängen, wird hier zunächst der Quadtree benutzt. Ob am Ende ein Binärbaum oder variable Anzahlen an Unterteilungen effizienter sind, ist eine Frage der Optimierung.
Im Quadtree kann zu einer Position (x, y) die richtige Zelle i aus den vier Nachfolgerzellen durch

$$i = \left\lceil \frac{x - x_0}{k_x} \right\rceil + 2 \cdot \left\lceil \frac{y - y_0}{k_y} \right\rceil \tag{4.1}$$

berechnet werden, wenn die Zellen wie in Abbildung 4.1 nummeriert sind.
Für die Verfolgung eines Teilchens durch die Zellen kann damit auch die Zelle bestimmt werden, die das Teilchen nach dem Bewegungsschritt erreicht hat. Zusätzlich muss aber bekannt sein, welche Zellen während der Bewegung überschritten werden, weil erst dadurch auch alle Randsegmente bekannt sind, mit denen ein Teilchen

während dieser Bewegung kollidieren könnte. Da nur rechteckige Zellen vorkommen, lassen sich auch dafür effiziente Methoden finden (vgl. Kap. 6.1).
Das Problem, dass ein Quadtree aus regulären Gittern nicht an Ränder angepasst werden kann, wird in Kapitel 5.1 weiter diskutiert.

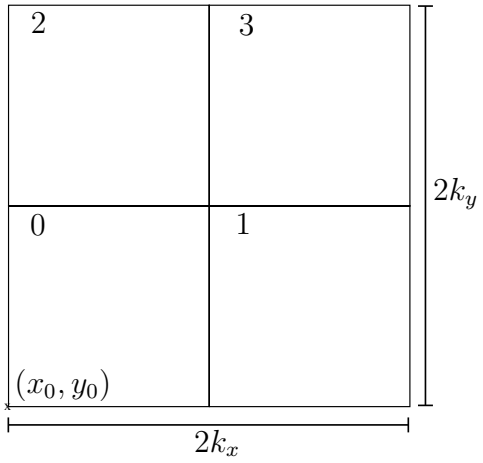

Abb. 4.1: Bezeichnungen im Quadtree für Gleichung 4.1

4.3 Voronoi-Diagramm

Die Delaunay-Triangulation ist ein etabliertes Verfahren zur Erzeugung von Rechengittern. Für DSMC würde sich aber das zur Delaunay-Triangulation duale Voronoi-Diagramm aufgrund seiner geometrischen Eigenschaften wesentlich besser eignen.
Benutzt man die euklidische Metrik, können nur konvexe Voronoi-Zellen entstehen (Beweis per Konstruktion; gilt z.B. nicht, wenn der Einfluss durch die Metrik der Supremumsnorm gegeben wird [16]).
Indirekte Nachbarn kommen nur sehr selten vor. Der Grund dafür ist, dass die Anzahl der Kanten, die sich in einem Knotenpunkt treffen, mit der Anzahl der Zentren, die auf einem Kreis liegen, und mit der Anzahl der Ecken des dualen Polygons im Delaunay-Gitter übereinstimmt. Diese Anzahl ist meistens drei, weil man drei Punkte immer mit einem Kreis verbinden kann, wenn diese nicht auf einer Gerade liegen. Bei mehr als drei Punkten ist dies in deutlich weniger Fällen möglich.
Ein besonders praktischer Vorteil von Voronoi-Diagrammen ist die Existenz inkrementeller Algorithmen zu ihrer Konstruktion. Dadurch kann die Adaption durch Hinzufügen und Entfernen von Zellen mit dem selben Algorithmus wie zur Kon-

struktion geschehen. So bleiben auch die geometrischen Eigenschaften eines Voronoi-Diagramms und damit die Qualität des Gitters beim Vergröbern oder Verfeinern erhalten.
Um aus einer vorgegebenen Menge von n Punkten das Voronoi-Diagramm zu berechnen, gibt es schnellere Algorithmen. Die inkrementelle Methode hat für diese Aufgabe die Zeitkomplexität $O(n^2)$, wohingegen Algorithmen der Sweep-Line- oder Divide-and-Conquer-Methode die Komplexität $O(n \log n)$ erreichen (eine Übersicht der wichtigsten Algorithmen gibt z.B. [1]). Die Komplexität der inkrementellen Methode setzt sich aus der Lokalisierung der Position einer neuen Zelle im alten Gitter ($O(n^2)$) und dem Einfügen der neuen Zelle ($O(n)$) zusammen. Bei der Adaption eines Rechengitters wird aber eine bestimmte Stelle im Gitter vorgegeben, an der die Netzdichte verändert werden soll. Damit entfällt eine zusätzliche Lokalisierung und für die Adaption ergibt sich nur noch ein zusätzlicher linearer Aufwand zur Konstruktion neuer Zellen.
Die inkrementelle Methode erzeugt das Voronoi-Diagramm über sein Delaunay-Gitter, weil die Überprüfung des Delaunay-Kriteriums einfacher ist als die Berechnung vieler Schnittpunkte. Für die Teilchenverfolgung hat dieser scheinbare Umweg aber den praktischen Vorteil, dass das Delaunay-Gitter nicht nur die Topologie des Voronoi-Diagramms wiedergibt, sondern gleichzeitig die Kanten des Delaunay-Gitters die Normalenvektoren der Voronoi-Kanten sind. Das Delaunay-Gitter ist also nicht nur einfacher zu erzeugen, es enthält genau die Informationen, die zur Teilchenverfolgung benötigt werden und aus einem direkten Gitter erst berechnet werden müssten.
Da das Voronoi-Diagramm für die Teilchenbewegung nicht gebraucht wird, könnte es nur bei einer Adaption punktuell für die betroffenen Zellen berechnet werden. Dies verursacht einen geringen zusätzlichen Rechenaufwand bei der Adaption, reduziert aber deutlich den Speicherbedarf des Gitters. Dieses Vorgehen könnte besonders im Dreidimensionalen von Bedeutung werden, weil das Voronoi-Diagramm dort wesentlich mehr (redundante) Informationen enthält als das Delaunay-Gitter. Die Ursache demonstrieren einige Werte [16] für so genannte Poisson-Voronoi-Diagramme, d.h. bei zufälliger Verteilung der Zentren. In der Ebene haben die Voronoi-Polygone im Durchschnitt höchstens 6 Ecken und Kanten und entsprechend hat die Zelle 6 Delaunay-Vektoren. In drei Dimensionen haben die Voronoi-Polygone im Durchschnitt schon etwa 27.1 Ecken und 40.6 Kanten, gebraucht werden aber nur die Delaunay-Vektoren der Flächen, von denen es im Durchschnitt etwa 15.5 gibt.

Da die Zentren aber nicht zufällig sondern durch einen Adaptionsalgorithmus verteilt werden sollen, lässt sich durch die Positionierung der Zentren Einfluss auf die entstehenden Polygone nehmen und es lässt sich eine große Vielfalt von Gittern erzeugen. In Abbildung 4.2 sind reguläre Gitter zu sehen, die sich durch regelmäßige Anordnung der Zentren ergeben.
In einem unstrukturierten Gitter kann der Adaptionsalgorithmus dafür sorgen, dass eine bestimmte Form bevorzugt entsteht. Dafür wird die Sechseckstruktur gewählt, weil diese dem für DSMC idealen Gitter am nächsten kommt (vgl. Kap. 3.3.1). Sie entsteht durch Zentren, die in den Mittelpunkten der Kreise in dichtester Packung liegen[1]. Um die regelmäßige Sechseckstruktur zu approximieren, sollte ein neues Zentrum also immer in den größten leeren Kreis, der kein anderes Zentrum enthält, platziert werden. Nützlicherweise kommen dafür nur die Umkreise der Delaunay-Dreiecke in Frage, d.h. ein neues Zentrum wird immer auf den Eckpunkt eines Voronoi-Polygons gesetzt.
Abbildung 4.4 zeigt ein nach dem eben skizzierten Vorschlag verfeinertes Gitter. Da eine neue Zelle von all ihren Nachbarzellen in etwa den selben Anteil Fläche bekommt, entsteht im Vergleich zu dem Poisson-Voronoi-Diagramm in Abbildung 4.3 eine kontinuierliche Änderung der Zellgrößen.

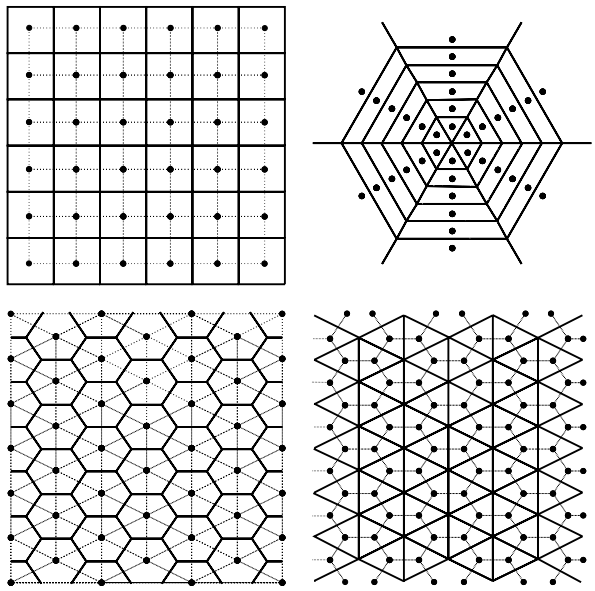

Abb. 4.2: Voronoi-Diagramme mit regelmäßigen Verteilungen der Zentren

[1] In 3D entstehen aus einer dichtesten Kugelpackung Dodekaeder.

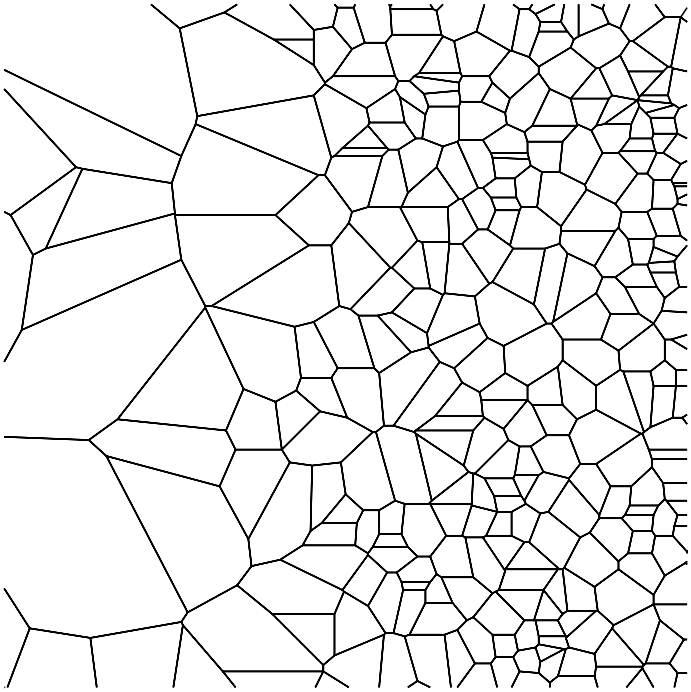

Abb. 4.3: Poisson-Voronoi-Diagramm mit zunehmender Netzdichte

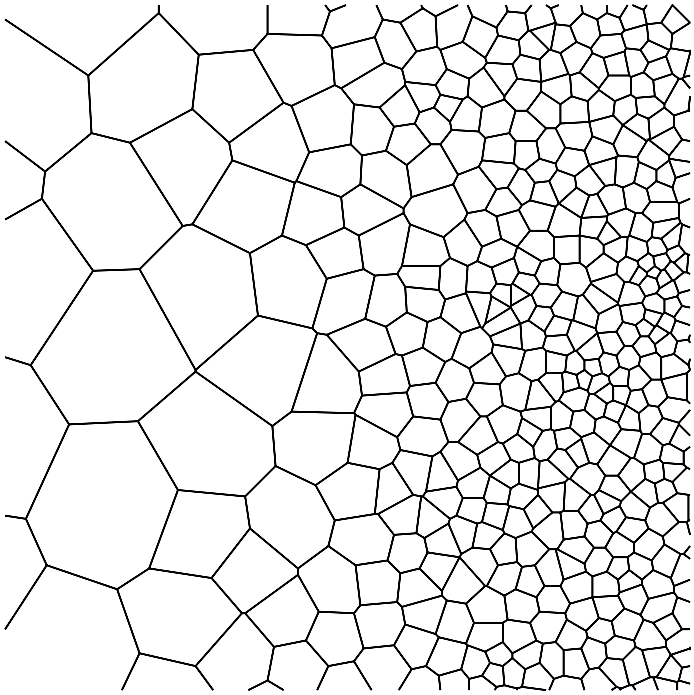

Abb. 4.4: Sechseckgitter approximierendes Voronoi-Diagramm mit zunehmender Netzdichte

5 Integration der Randkontur

5.1 Quadtree

Da die rechteckigen Zellen des Quadtrees nicht an beliebige Ränder angepasst werden können (vgl. Kap. 3.3.1), muss die Randkontur in anderer Weise berücksichtigt werden.
Bei jeder Bewegung eines Teilchens die Randkontur direkt auf Kollisionen zu testen wäre zu aufwändig, weil die Kontur aus einer großen Anzahl von Randsegmenten bestehen kann. Außerdem sind Schnittpunkte zwischen der Flugbahn und einem Randsegment deutlich aufwändiger zu berechnen, als Schnittpunkte der Flugbahn mit den Kanten einer konvexen Zelle in einem randangepassten Gitter.
In den meisten Fällen kommt aber der größte Teil der Randsegmente für eine Kollision gar nicht in Frage. Deshalb soll mit Hilfe des Quadtrees für jede Bewegung eine möglichst kleine Menge von Randsegmenten ausgewählt werden, welche dem Teilchen im Weg sein könnten. In dieser Sicht ist der Quadtree kein wie in Kapitel 3.4.2 beschriebenes Rechengitter, sondern dient dazu, ein Gitter so weit wie möglich zu vermeiden. Dafür werden die Randsegmente so in den Quadtree einsortiert, dass zu jeder Zelle bekannt ist, welche Randsegmente diese überdeckt. Dann werden bei einer Teilchenbewegung nur noch die Randsegmente überprüft, welche von durchquerten Zellen überdeckt werden. Dies wird in Kapitel 6.1 weiter beschrieben.
Die Auswertung des aktuellen Strömungszustandes und die Teilchenverfolgung müssen nicht auf der selben Stufe des Quadtrees arbeiten. Der Quadtree kann daher für die Teilchenbewegung so weit eingeteilt werden, dass die Ausdehnung der Zellen kleiner als die mittlere freie Weglänge der Teilchen ist und so möglichst wenig Randsegmente überprüft werden müssen. Nur die Auswertung benötigt noch eine konvexe Zerlegung der Zellen. Dafür sollte der Quadtree mindestens so weit eingeteilt werden, dass eine Zelle maximal eine Ecke enthält, an der sich die Randsegmente nicht konvex treffen (vgl. Abb. 5.1 links). Eine konvexe Zerlegung dieser Zellen kann dann durch ein einfaches Verfahren bestimmt werden, wie schon in Kapitel 3.4.2 beschrieben

wurde. In Abbildung 5.1 rechts ist ein innerer Rand dargestellt, der zu einer Zerlegung in sehr spitzwinkeligen Zellen führt. Die ursächliche Einteilung des Quadtrees lässt sich auch nicht verhindern, weil der Quadtree für die beiden nicht-konvexen Ecken oben rechts weiter eingeteilt werden muss. Das Beispiel demonstriert, wie im Quadtree für die Auswertung unvermeidlich Zellen beliebig schlechter Qualität entstehen können.

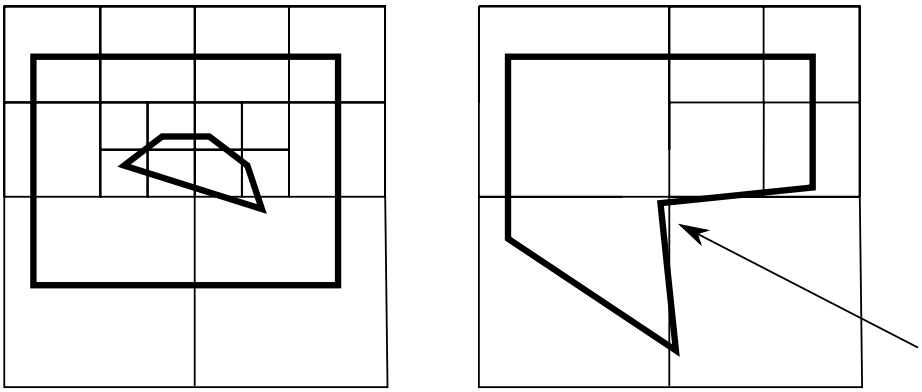

Abb. 5.1: Links: Innerer und äußerer Rand im Quadtree in der Einteilung für die Auswertung
Rechts: Ein innerer Rand, bei dem die Einteilung sehr spitzwinkelige Zellen erzeugt

5.2 Voronoi-Diagramm

Alle bisher gezeigten Abbildungen von Voronoi-Diagrammen zeigen nur einen Ausschnitt, weil die Zellen der äußeren Zentren, welche die konvexe Hülle aller Zentren bilden, keine endliche Ausdehnung haben. Damit aus einem Voronoi-Diagramm ein Voronoi-Rechengitter wird, müssen die Zellen irgendwie durch äußere und innere Ränder begrenzt werden können und dabei für die Adaption so dynamisch wie möglich bleiben.
Eine einfache Idee Ränder abzubilden wäre, die Zentren an den Rändern zu spiegeln, um damit wie in der Potentialtheorie üblich Wände zu erzeugen. Jedes gespiegelte Zentrum könnte aber die Symmetrie an einer anderen, schon bestehende Wand wieder zerstören. Wenn diese nach der selben Methode korrigiert würde, könnte sich der Defekt endlos fortsetzen.

Wenn man andererseits die Zentren völlig unabhängig vom Rand verteilt, müssen die Zellen gemäß den Randsegmenten, die sie gerade überdecken, zerlegt werden. Bei nur wenigen Zentren im Vergleich zur Anzahl der Randsegmente würde diese Zerlegung kompliziert, weil eine Zelle sehr viele Randsegmente überdecken kann.
Bei beiden Methoden müsste ständig darauf geachtet werden, dass in jeder Region ausreichend viele Zellen vorhanden sind, um den Rand vernünftig abzubilden. Am einfachsten lässt sich dies erreichen, indem z.B. auf den Ecken oder dem Rand einige statische Zentren platziert werden, die nicht entfernt werden können. Diesen Weg verfolgen auch die zwei wichtigsten bekannten Verfahren [1], mit denen eine Delaunay-Triangulationen an ein berandetes Gebiet angepasst wird.
Bei der Constrained-Delaunay-Triangulation wird für einen Rand R die euklidische Metrik $d(\vec{x}, \vec{y})$ ersetzt durch

$$b(\vec{x}, \vec{y}) = \begin{cases} d(\vec{x}, \vec{y}), & \text{falls } \overline{xy} \cap R = \emptyset \\ \infty & \text{sonst} \end{cases} . \tag{5.1}$$

Die Voronoi-Polygone werden so auf den Bereich beschränkt, der von ihrem Zentrum aus sichtbar ist. In der Regel wird auf jede Ecke der Randkontur ein Zentrum platziert, weil dadurch die Bedingung der Sichtbarkeit einfacher kontrolliert werden kann. Eine offensichtliche Folge aus der verwendeten Metrik ist, dass am Rand nichtkonvexe Voronoi-Polygone entstehen (vgl. Abb. 5.2).

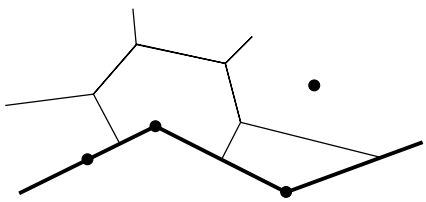

Abb. 5.2: Voronoi-Diagramm einer Constrained-Delaunay-Triangulation

Weniger einsichtig ist das Problem, dass man nur schwer sicherstellen kann, dass durch dieses Voronoi-Diagramm das gesamte Rechengebiet abgedeckt wird und keine Löcher im Gitter entstehen. Die Situation entspricht dem so genannten Art-Gallery-Problem: wie viele Wächter braucht man, um alle Wände einer Kunstgalerie zu bewachen, wenn die Wächter nicht umher gehen, aber in jede Richtung sehen können? In 2D ist eine nicht optimale aber sehr einfache Lösung, in jeder Ecke des Polygons einen Wächter zu platzieren. Aber in 3D funktioniert diese Lösung nicht, weil es in einem Polyeder Orte geben kann, von denen aus keine der Ecken sichtbar ist [17].
Im Gegensatz dazu benutzt eine Conforming-Delaunay-Triangulation weiterhin die euklidische Metrik. Zunächst wird auf jede Ecke des Polygons ein Zentrum gesetzt.

Wenn dann ein Teil des Randes nicht Teil der Delaunay-Triangulation ist, werden an entsprechenden Stellen auf dem Rand Steiner-Punkte hinzugefügt, bis der Rand durch die Triangulation abgebildet wird.
Bei Rändern mit spitzen Winkeln taucht auch hier wieder das Problem auf, dass sich ein Defekt beliebig weit fortsetzen kann. Dies lässt sich mit einer Methode von Ruppert [19] verhindern, indem für den ersten Steiner-Punkt die Mitte der fehlenden Kante und für alle weiteren der dem neuen Mittelpunkt am nächsten liegende Schnittpunkt gedachter Kreise mit quadratischen Abständen um den Eckpunkt gewählt wird (vgl. Abb. 5.3 rechts). In 2D ist das Problem damit gelöst, aber in 3D

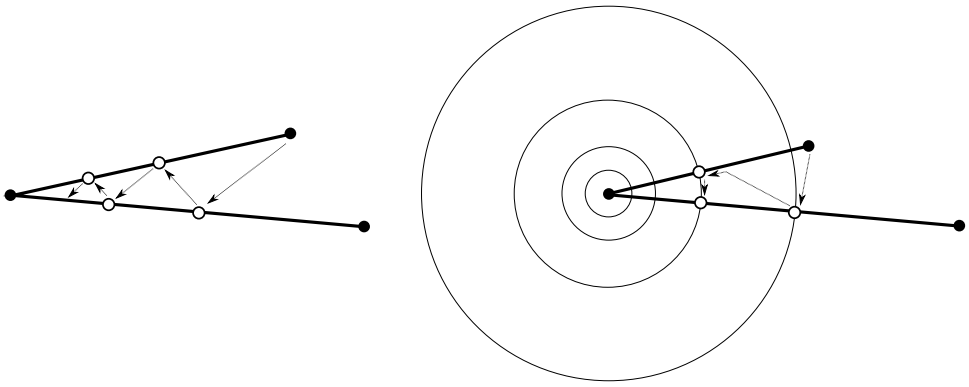

Abb. 5.3: Das Konvergenz-Problem bei kleinen Winkeln und Rupperts Lösung

müssten jetzt noch die Flächen abgebildet werden. Wenn die Kanten einer Fläche durch die Triangulation abgebildet werden, bedeutet das noch nicht, dass auch die Fläche zwischen diesen Kanten mit den Flächen der Tetraeder übereinstimmt. Das Problem der Konvergenz lässt sich für zwei Flächen mit spitzem Winkel nicht so leicht lösen wie für zwei Kanten mit spitzem Winkel.
Um auf sichere Art eine Triangulation zu erzeugen, kann nach einer limitierten Anzahl von Steiner-Punkten abgebrochen werden und die Triangulation vom Delaunay-Kriterium abweichend vervollständigt werden [4]. Für ein Voronoi-Rechengitter ist das keine Möglichkeit, weil ohne Delaunay-Kriterium auch keine korrekten Voronoi-Polygone entstehen.
Eine dynamische Zerlegung einer Zelle nach den Randsegmenten, die ihr Voronoi-Polygon gerade überdeckt, wäre geeignet, wenn die Zelle nur durch Flächen begrenzt werden soll, führt aber zu Problemen bei mehreren Ecken und Kanten. Die Conforming-Delaunay-Triangulation funktioniert für Ecken und Kanten einwandfrei, hat aber Probleme damit, Flächen abzubilden. Beide lassen sich zu einer ver-

gleichsweise einfachen Methode kombinieren, um aus einem Voronoi-Diagramm ein an beliebig komplizierte Polyeder angepasstes Voronoi-Rechengitter zu erzeugen.
In 2D wird auf jede Ecke des Randes ein festes Zentrum gesetzt. In 3D werden zusätzlich die Kanten, jedoch nicht die Flächen, als eine vereinfachte Conforming-Delaunay-Triangulation abgebildet. Von den Kanten in 2D bzw. den Flächen in 3D muss nur noch bekannt sein, von den Voronoi-Polygonen welcher Zellen diese überdeckt werden. Aus der Delaunay-Triangulation allein kann dann wie üblich ein Voronoi-Diagramm abgelesen werden. Das Voronoi-Rechengitter wird jedoch zusammen aus der Delaunay-Triangulation und dem Rand abgelesen und jede Änderung von einem der beiden übernommen.
Es gibt also feste Zentren an den Eckpunkten des Randes, freie Zentren, welche beliebig für die Adaption verwendet werden dürfen, und in 3D noch die Zentren der Steiner-Punkte auf den Kanten, welche automatisch erzeugt werden und nur dann zur Adaption gelöscht werden dürfen, wenn dabei die Kante in der Delaunay-Triangulation erhalten bleibt.
In 2D wird jeder Eckzelle, deren beider Randsegmente keine konvexe Ecke bilden, zusätzlich die Winkelhalbierende der Randsegmente als ein für Teilchen durchlässiger Rand zugewiesen. In 3D werden den Steiner-Zentren[1] auf den Kanten entsprechend die winkelhalbierenden Ebenen zugewiesen. Für die Ecken in 3D lässt sich nicht ganz so leicht eine konvexe Zerlegung konstruieren. Glücklicherweise wurde diese Aufgabe bereits durch das Delaunay-Gitter gelöst. Die Eckzellen bekommen also zusätzlich von den Delaunay-Tetraedern, die diese Ecke berühren, nur die Flächen, die diese Ecke berühren. Dazu wird am besten die minimale Delaunay-Triangulation (nur die Zentren auf den Ecken und den Steiner-Punkten) benutzt, weil sie die wenigsten Tetraeder enthält, und wenn möglich sollten daraus noch Flächen gelöscht werden, solange dabei die Zerlegung konvex bleibt.
Diese Konstruktion steht fest, solange sich der Rand nicht verändert, und wird von den Randsegmenten, die zusätzlich gerade von der Voronoi-Zelle überdeckt werden, ergänzt. Erst durch das veränderliche Voronoi-Diagramm wird diese bis jetzt offene Zerlegung wie in Abbildung 5.4 zu konvexen Zellen des Voronoi-Rechengitters geschlossen.
Da die Winkelhalbierende wie die gestrichelte Linie in Abbildung 5.4 links meist auf eine Voronoi-Kante trifft, ist hier die Topologie nicht eindeutig, weil eine Zelle mehr Nachbarzellen als Kanten hat. Statt der Winkelhalbierenden eine Verbindungslinie

[1] Einer Constrained-Delaunay-Triangulation würden diese notwendigen Zentren fehlen.

zwischen der Ecke und einem der beiden Punkte $\vec{p}_1$ oder $\vec{p}_2$ als durchlässigen Rand zu verwenden, wird in 2D nicht immer eine konvexe Zerlegung ergeben. In 3D ist so eine Verbindung in der Regel gar nicht möglich. In 2D lässt sich das Problem leicht lösen, indem jede Kante, an die zwei Nachbarzellen grenzen, wie in Abbildung 5.4 rechts, in zwei nicht parallele Kanten geteilt wird. In 3D müssten dabei die Eckzellen in viele weitere Zellen geteilt werden, weil jede Fläche mit zwei Nachbarzellen, die von k Kanten begrenzt wird, entsprechend in k Flächen zerlegt werden müsste und dabei die Konvexität der Eckzellen verloren ginge. Deshalb wird hier auch auf die einfache Lösung in 2D verzichtet und das Problem bei der Teilchenverfolgung im Kapitel 6.2 weiter behandelt.
Das Ergebnis des beschriebenen Voronoi-Rechengitters ist beispielhaft mit einem Quadrat als äußeren Rand und einem weiteren Polygon als inneren Rand in den Abbildungen 5.5 und 5.6 zu sehen.

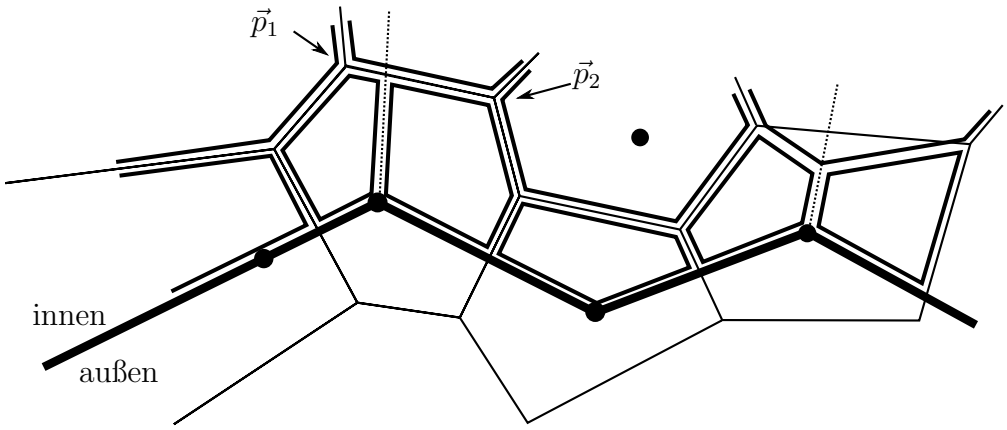

Abb. 5.4: Ränder (dicke, durchgezogene Linie) und Winkelhalbierende (gestrichelte Linien) stehen fest, alle anderen Kanten verändern sich mit dem Voronoi-Diagramm. Die Lösung rechts funktioniert nur in 2D.

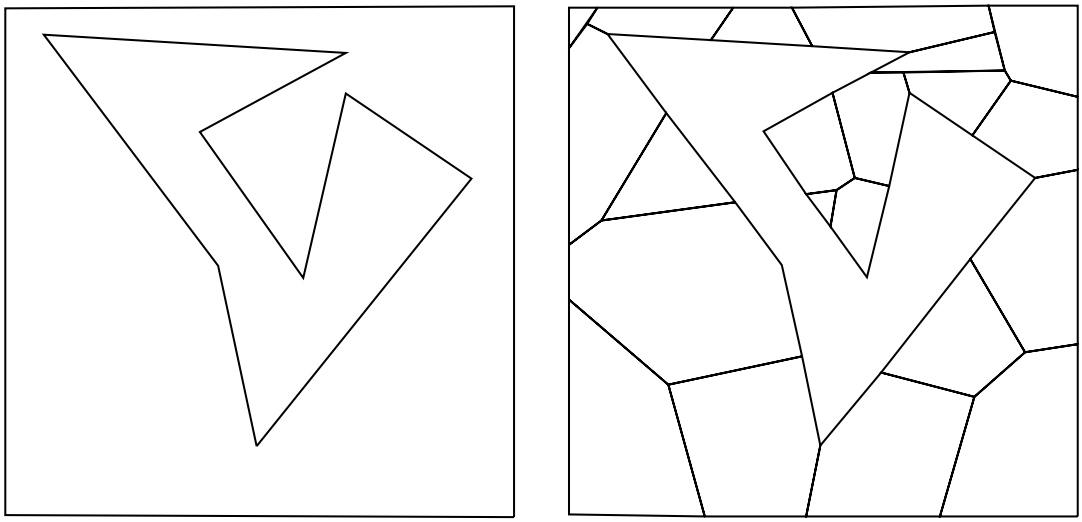

Abb. 5.5: Links: Randkontur aus einem Quadrat und einem weiteren Polygon
Rechts: Minimales Gitter

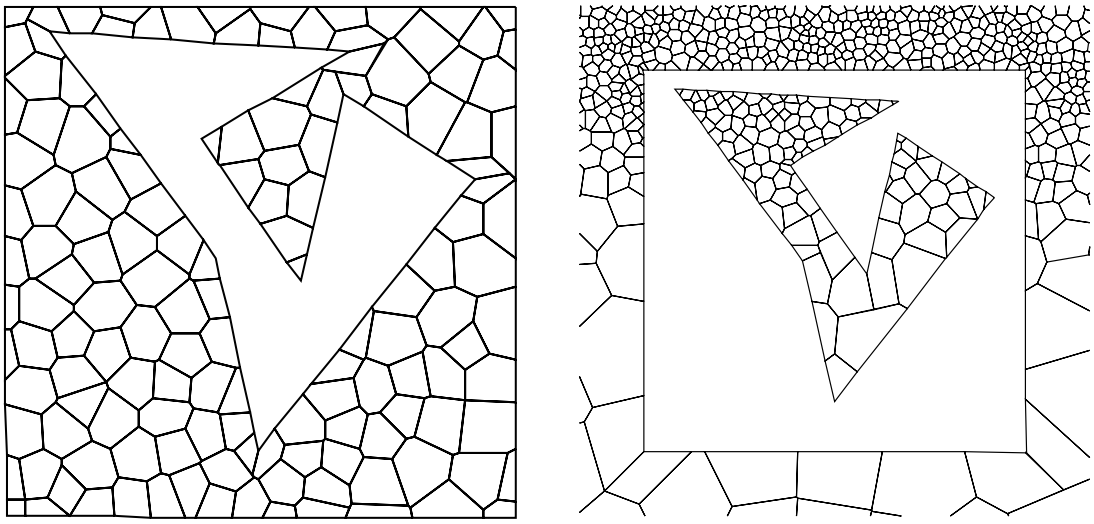

Abb. 5.6: Links: Gleichmäßig adaptiertes Gitter
Rechts: Ungleichmäßig adaptiertes Gitter bei umgekehrten Rändern

6 Teilchenverfolgung

6.1 Quadtree

Bei der Bewegung eines Teilchens im Quadtree sollen die Randsegmente des Rechengebietes auf Kollision überprüft werden. Dazu müssen die Schnittpunkte zwischen der Flugbahn und einer Menge von Randsegmenten berechnet werden. Das Teilchen kollidiert mit dem Rand, wenn der Schnittpunkt der kleinsten Entfernung innerhalb der Bewegung liegt.
Da die Randkontur, wie in Kapitel 5.1 beschrieben, in den Quadtree einsortiert ist, müssen nur noch die Zellen gefunden werden, die das Teilchen während seiner Bewegung durchqueren würde. Statt alle Zellen daraufhin zu testen, kann von der Wurzel oder vom gemeinsamen Vorgänger der Start- und Zielzelle abwärts gesucht werden.

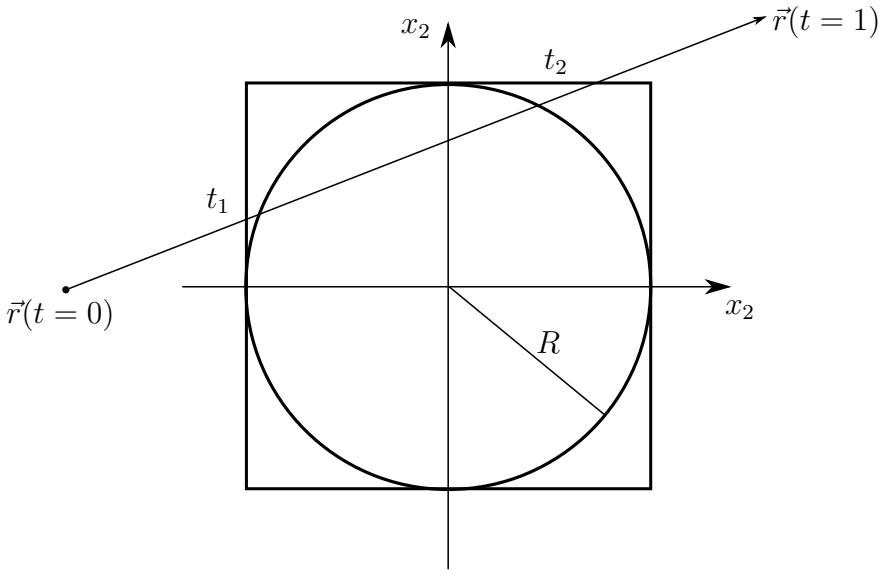

Abb. 6.1: Kreis mit Radius R um den Ursprung in euklidischer Norm und Supremumsnorm

Die Bewegung des Teilchens sei $\vec{r}(t) = \vec{r}_\mathrm{p} + t\vec{v}$ mit $t \in T := [0;1]$. Angenommen, die Zellen hätten die Form eines Kreises mit Radius R um den Ursprung des Koordinatensystems, dann würde das Teilchen den Kreis durchqueren, wenn ein t existiert, für das

$$\|\vec{r}(t)\| \leq R \quad . \tag{6.1}$$

Unter Verwendung der Supremumsnorm sieht der Kreis mit Radius R aus wie das Quadrat mit Kantenlänge $2R$ in euklidischer Norm. Die Idee lässt sich direkt übertragen.
Es wird das Intervall $I = [t_-; t_+]$ bestimmt, für das sich das Teilchen innerhalb des Quadrates aufhält. Für jede Koordinate k lässt sich das Intervall I_k aus der Bedingung 6.1 berechnen:

$$I_k = [t_{-,k}; t_{+,k}] \quad \text{mit} \quad t_{\pm,k} = \frac{\pm R - r_{\mathrm{p},k}}{v_k} \quad . \tag{6.2}$$

Das Quadrat wurde dann durchquert, wenn

$$I = T \cap \bigcap_{k=1}^{2,3} I_k \neq \emptyset \quad . \tag{6.3}$$

Wenn schon $T \cap I_1 = \emptyset$ müssen die anderen Intervalle erst gar nicht berechnet werden.

6.2 Voronoi-Diagramm

Als Grundlage für die Teilchenverfolgung im Voronoi-Gitter soll hier kurz eine sehr robuste Methode zur Teilchenverfolgung in einem Gitter konvexer Zellen skizziert werden, die von A. Haselbacher et. al. [7] vorgestellt wurde.
Die Beschreibung erfolgt in der Ebene, lässt sich aber direkt auf drei Dimensionen erweitern, indem die Kanten durch Flächen ersetzt werden. Im Folgenden wird eine Bewegung auf die Zeit zwischen $t = 0$ und $t = 1$ normiert.
Ausgehend von der Zelle, in der sich das Teilchen aktuell befindet, wird es von Zelle zu Zelle verfolgt. Dies wird durch die Konvexität der Zelle vereinfacht, weil statt der Schnittpunkte mit den Kanten nur die Entfernung bzw. die Zeit bis zum Auftreffen auf den Kanten berechnet werden muss. Ist die Zeit bis zum ersten Auftreffen kleiner als 1, wird diese Kante noch innerhalb des Bewegungsschrittes durchquert und der

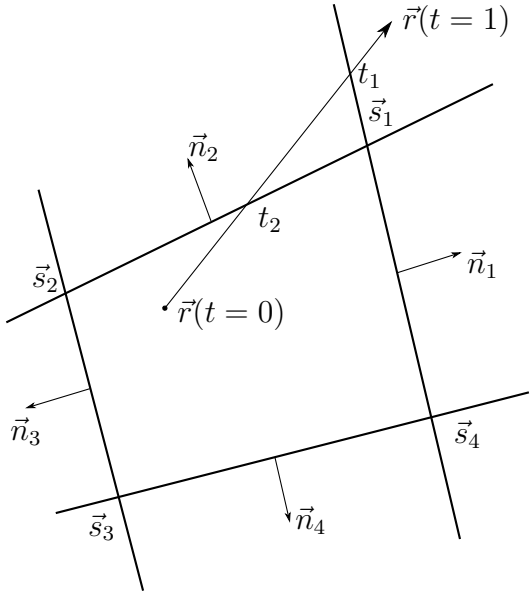

Abb. 6.2: Bestimmung der Schnittpunkte in der Zelle

Rest des Weges in der gegenüberliegenden Zelle fortgesetzt.
Alle Kanten i werden nur durch ihren Normalvektor $\vec{n}_i$ und einen Aufpunkt $\vec{s}_i$ als unbegrenzte Geraden beschrieben. Die geradlinige Bewegung eines Teilchens ist durch dessen Startposition $\vec{r}_\mathrm{p}$ und Geschwindigkeit $\vec{v}$ gegeben.
Wenn die Normalvektoren einer Zelle nach außen zeigen, bewegt sich das Teilchen auf eine Kante i zu, wenn $\vec{v} \cdot \vec{n}_i > 0$. Für alle Kanten i, die diese Bedingung erfüllen, wird die Zeit bis zum Auftreffen berechnet:

$$\begin{aligned} \text{Kante:} \quad & (\vec{r} - \vec{s}_i) \cdot \vec{n}_i = 0 \\ \text{Bewegung:} \quad & \vec{r}(t) = \vec{r}_\mathrm{p} + t\vec{v} \text{ für } 0 \leq t \leq 1 \\ \Rightarrow \quad & t_i = \frac{(\vec{s}_i - \vec{r}_\mathrm{p}) \cdot \vec{n}_i}{\vec{v} \cdot \vec{n}_i} \quad . \end{aligned} \tag{6.4}$$

Falls ein Teilchen einer falschen Zelle zugeordnet ist, wird dieser Fehler erkannt, indem zusätzlich bei jeder Bewegung getestet wird, ob sich das Teilchen tatsächlich in der gegebenen Zelle befindet. Das Teilchen ist außerhalb der Zelle, wenn es eine Kante i gibt, für die $(\vec{s}_i - \vec{r}_\mathrm{p}) \cdot \vec{n}_i < 0$. Die Kante i, die zuerst überschritten wird, und die Zeit t_i bis dahin kann damit durch Algorithmus 1 bestimmt werden, selbst wenn die Teilchen völlig falsch zugeordnet sind.

Alg. 1 Teilchenverfolgungsalgorithmus von Haselbacher

```
t_min ← 1
for all i do
  if (s_i − r_p) · n_i ≥ 0 then
    if v · n_i > 0 then
      t_i ← ((s_i − r_p) · n_i) / (v · n_i)
      if t_i < t_min then
        t_min ← t_i
        i_min ← i
      end if
    end if
  else
    t_min ← 0
    i_min ← i
    exit
  end if
end for
```

Durch numerische Fehler kann es immer vorkommen, dass ein Teilchen nicht der Zelle zugeordnet ist, in der es sich geometrisch befindet. Im Voronoi-Gitter ist es außerdem nicht sinnvoll, bei jeder Adaption des Gitters die Teilchen aller betroffenen Zellen zu kontrollieren, weil ein großer Teil der Teilchen noch korrekt zugeordnet sein kann. Der Teilchenverfolgungsalgorithmus für das Voronoi-Gitter sollte unbedingt Fehler korrigieren können, weil dann auch bei der Adaption auftretende Fehlzuordnungen ignoriert werden können und so bei der nächsten Bewegung nur die Teilchen korrigiert werden, die wirklich falsch zugeordnet waren. Damit wäre außerdem das in Kapitel 5.2 beschriebene Problem der nicht eindeutigen Nachbarschaft an den Eckzellen gelöst, weil dann Teilchen, die eine Kante mit zwei Nachbarzellen durchqueren, beliebig einer von beiden Nachbarzellen zugeordnet werden können.
In dem Algorithmus von Haselbacher werden Fehlzuordnungen korrigiert, indem die Entfernung zu einer Kante nur dann berechnet wird, wenn sich das Teilchen vor dieser Kante befindet. Wenn das Teilchen hinter der Kante liegt ist es damit außerhalb der Zelle und das Teilchen wird nicht bewegt, sondern der gegenüberliegenden Zelle zugeordnet. Anschaulich gesprochen wird jedes Teilchen erst von der Ausgangszelle zur Zelle der Startposition verfolgt und von dieser aus zur Zelle der Zielposition.
Wären keine Wände im Weg, würde es genügen, das Teilchen von der möglicherweise falschen Zelle nur zu seinem Ziel zu verfolgen. Dafür wird für jede Kante getestet, ob das Ziel hinter der Kante liegt und nur dann die Zeit bis zum Überqueren dieser Kante bestimmt. Wenn das Teilchen die Zelle gar nicht verlässt, werden die Zeiten

auch nicht benötigt.
Die Kante i, die zuerst überschritten wird, und die Zeit t_i bis dahin kann nach dieser Idee durch Algorithmus 2 bestimmt werden. Eine Fehlzuordnung liegt genau dann vor, wenn zu einer Kante i eine Zeit $t_i < 0$ oder $t_i > 1$ berechnet wird.

Alg. 2 Vorgeschlagener Algorithmus

```
t_min ← 1
for all i do
  if (r(t = 1) − s_i) · n_i ≥ 0 then
    if v · n_i ≠ 0 then
      t_i ← ((s_i − r_p) · n_i) / (v · n_i)
    else
      t_i ← ∞
    end if
    if t_i < t_min then
      t_min ← t_i
      i_min ← i
    else if t > 1 then
      t_min ← 0
      i_min ← i
    end if
  end if
end for
if t_min < 0 then
  t_min ← 0
end if
```

Die Behandlung der Kanten einer einzelnen Zelle wird schneller, da die Bedingung, dass das Teilchen nach seiner Bewegung hinter einer Kante liegt, seltener erfüllt ist als die Bedingung, dass das Teilchen vor seiner Bewegung vor der Kante liegt. Außerdem wird die Korrektur falsch zugeordneter Teilchen schneller. In Abbildung 6.3 links ist dazu eine Beispielsituation dargestellt, die nicht selten auftreten wird, wenn eine neue Zelle in das Gitter eingefügt wird. Durch die neue Zelle B befindet sich das Teilchen außerhalb seiner ursprünglichen Zelle A, wird aber bei der ersten Bewegung wieder in diese zurückkehren. Der Algorithmus 1 wird in dieser Situation alle Kanten von A testen, den Fehler bemerken, das Teilchen Zelle B zuweisen, alle Kanten von B testen und, da Zelle B verlassen wird, das Teilchen wieder Zelle A zuordnen und hier ein weiteres Mal alle Kanten prüfen. Der Algorithmus 2 wird nur die Kanten von A prüfen und dabei feststellen, dass das Teilchen nach seiner

Bewegung korrekt zugeordnet ist.
Angenommen, das Teilchen ist falsch zugeordnet und trifft währen seiner Bewegung auf eine Wand. Eine Beispielsituation an einer Ecke ist in Abbildung 6.3 rechts dargestellt. Das Teilchen verlässt Zelle C durch die Kante, an die zwei Nachbarzellen grenzen, und wird fälschlicherweise Zelle D zugeordnet. Mit Algorithmus 1 wird der Fehler erkannt. Das Teilchen wird Zelle E zugeordnet und kollidiert mit der Wand. Mit Algorithmus 2 überspringt das Teilchen die Wand, weil der Fehler nicht erkannt wird, und liegt dann in Zelle D.
Der beschriebene Fehler kann nur an den Ecken der Randkontur zu Problemen führen, weil sich auch bei Adaption nicht ändern kann, ob eine Zelle vor oder hinter einer Wand liegt. Deshalb genügt es, eine zusätzliche Kontrolle, ob das Teilchen richtig zugeordnet ist, nur innerhalb der Eckzellen oder sogar nur für Teilchen, die von bestimmten Zellen (im Beispiel C) in eine Eckzelle gelangen, einzusetzen.
Ein Laufzeitvergleich der beiden Algorithmen wird in Tabelle 7.1 im folgenden Kapitel 7 gegeben.

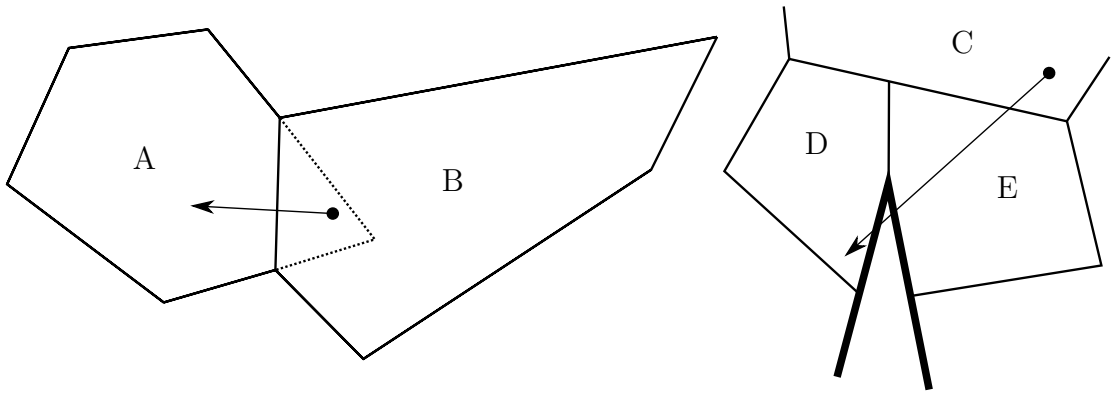

Abb. 6.3: Links: Zelle B ist neu hinzugekommen. Das Teilchen ist fälschlicherweise noch Zelle A zugeordnet und bewegt sich entlang des Pfeils.
Rechts: Das Teilchen überquert die Kante mit zwei Nachbarzellen und wird fälschlicherweise Zelle D zugeordnet.

7 Implementierung und Effizienz

Die Implementierung eines Quadtrees ist ein Standardwerkzeug der Programmierung und die Teilchenverfolgung enthält auch in 3D keine über das Kapitel 6.1 hinausgehende Besonderheiten. Auf die Zerlegung der Ecken der Randkontur in konvexe Zellen wurde verzichtet, weil dies für die Teilchenverfolgung nicht benötigt wird.
Die Datenstruktur des Voronoi-Diagramms ist ein einfacher Graph, dessen Verbindungen zwischen den Zentren die Delaunay-Triangulation repräsentieren. Zu jedem Zentrum werden die entsprechenden Zellen des Voronoi-Rechengitters gespeichert. Bei jeder Veränderung in der Delaunay-Triangulation wird auch das Voronoi-Rechengitter angepasst. Das Voronoi-Diagramm wird wie in Kapitel 4.3 beschrieben nur zur Adaption und zur graphischen Ausgabe berechnet.
Die inkrementelle Konstruktion des Delaunay-Gitters kennt nur Hinzufügen und Entfernen von Zentren auf einem bestehenden Gitter. Deshalb wird eine initiale Delaunay-Triangulation aus mindestens drei Zentren benötigt. Da die Änderungen nur innerhalb dieser ersten Triangulation geschehen kann und damit die maximale Ausdehnung des Rechengebietes definiert ist, werden dafür gleich vier Zentren in quadratischer Anordnung benutzt. Erst in diese initiale Delaunay-Triangulation wird die Randkontur eingebaut. Als Beispiel sind in Abbildung 7.1 noch einmal das minimale Voronoi-Rechengitter der Randkontur aus Kapitel 5.2, das Voronoi-Diagramm und die vier Zentren der initialen Delaunay-Triangulation abgebildet.
Die Algorithmen für das Hinzufügen und Entfernen sind aus einem Artikel von M. A. Mostafavi et. al. [15] entnommen. Ein Zentrum wird hinzugefügt, indem es mit den drei Zentren verbunden wird, in deren Delaunay-Dreieck es liegt. Dann wird diese (meist falsche) Triangulation mit Hilfe des Umkreiskriteriums lokal korrigiert. Beim Entfernen eines Zentrums werden in dem entstehenden Loch unter den ehemaligen Nachbarn mögliche Dreiecke gesucht, die das Umkreiskriterium erfüllen, und die so ermittelten Nachbarn verbunden, bis kein gültiges Dreieck mehr gefunden werden kann.
Die Erweiterung des benutzten Algorithmus für das Hinzufügen von Zentren in drei

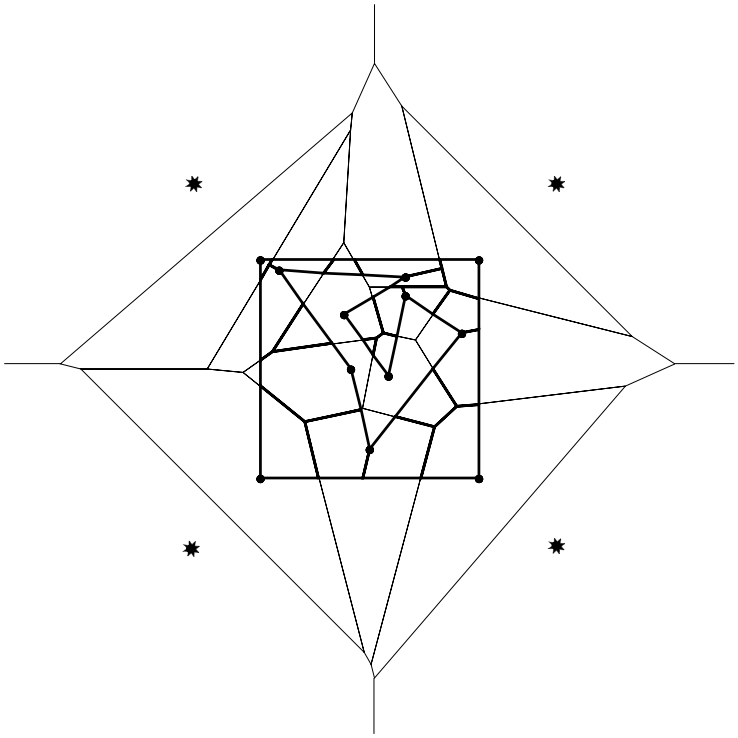

Abb. 7.1: Beziehung zwischen Voronoi-Rechengitter und Voronoi-Diagramm. Die sternförmigen Zentren werden als initiale Delaunay-Triangulation vorgegeben.

Dimensionen wird in [9] beschrieben, der benutzte Algorithmus zum Entfernen ist eine 2D-Variante des Algorithmus von Devillers [5]. In drei Dimensionen wird zusätzlich noch die Conforming-Delaunay-Triangulation [4] in der in Kapitel 5.2 beschriebenen vereinfachten Variante benötigt.
Bei jedem Algorithmus für Delaunay-Triangulationen beliebiger Verteilung von Zentren stellt sich das Problem, wie mit mehreren Zentren auf einer Linie oder auf einem Kreis umgegangen wird. Dazu gibt es eine Reihe von Vorschlägen [20]. Da die Zentren für das Voronoi-Rechengitter aber nicht beliebig, sondern in den Umkreis eines Delaunay-Dreiecks gesetzt werden, sind sie weit von diesen Problemfällen entfernt. Nur im minimalen Gitter können Probleme auftreten, beispielsweise wenn die Randkontur ein Rechteck enthält. Deshalb wurde dem Delaunay-Kriterium eine kleine Toleranz gegeben.
Als Teilchenverfolgungsalgorithmen wurden die beiden in Kapitel 6.2 beschriebenen Algorithmen implementiert, so dass diese direkt miteinander verglichen werden können.

Beide Konzepte sind komplett in Python geschrieben. Dies vereinfacht die Entwicklung, ist aber in der Ausführungsgeschwindigkeit nicht optimal. Für sehr rechenintensive Algorithmen wie die Teilchenverfolgung gibt es effizientere Sprachen. Da die Rechenzeit zusätzlich noch durch die verwendete Hardware bestimmt wird, sind die Verhältnisse der Messwerte von größerem Interesse als die absoluten Werte.
Im Voronoi-Rechengitter haben die Ränder kaum Einfluss auf die benötigte Rechenzeit zur Teilchenverfolgung, weil die Randsegmente gleichzeitig Kanten einer Zelle sind. Deshalb kann als Beispielsituation ein beliebiges Rechengitter ohne Ränder benutzt werden. Für die Messung wurden 200 Teilchen entlang der Linie in Abbildung 7.2 durch genau 30 Zellen bewegt. Im idealen Gitter (vgl. Kap. 3.1) würden diese auch in etwa 30 Zeitschritten durchquert, sodass die mittlere freie Weglänge der Ausdehnung der Zellen entspricht. In Tabelle 7.1 sind die Rechenzeiten für 15, 30 und 60 Zeitschritte der beiden in Kapitel 6.2 vorgestellten Algorithmen angegeben. Der Algorithmus 2 benötigt im Vergleich zu Algorithmus 1 nur etwa die Hälfte der Rechenzeit. Bei 30 Schritten ist in Klammern zusätzlich die Zeit angegeben, die bei einer willkürlichen Fehlzuordnung pro Bewegung entsteht. Dazu wurden die Teilchen nach jedem Schritt von ihrer aktuellen Zelle einer beliebigen Nachbarzelle zugeordnet. Dies hat die Rechenzeit bei Algorithmus 1 fast verdoppelt und bei Algorithmus 2 nur um die Hälfte erhöht.
Ein Vergleich mit der Teilchenverfolgung im Quadtree ist sehr schwierig, weil hier der Rand eine große Rolle spielt. Weil es im Voronoi-Gitter kaum einen Unterschied macht, ob die Teilchen sich frei bewegen oder bei jedem Schritt mit einer Wand kollidieren, sollen diese beiden Fälle den Bereich eingrenzen, über den sich die Rechenzeit der Teilchenverfolgung im Quadtree erstrecken kann. Die Rechenzeiten in Tabelle 7.1 beinhalten allerdings noch nicht die Zeit, die zum Einsortieren der Teilchen nach der Bewegung in die Zellen der Auswertung benötigt wird.

	15 Schritte	30 Schritte (mit Fehler)	60 Schritte
Voronoi-Rechengitter:			
Algorithmus 1	4,572 s	6,228 s (11,285 s)	9,057 s
Algorithmus 2	2,584 s	2,696 s (3,960 s)	3,664 s
Quadtree:			
Minimum	0,66 s	1,32 s	2,64 s
Maximum	14,25 s	29,04 s	58,08 s

Tab. 7.1: Gemessene Rechenzeit für die Bewegung von 200 Teilchen

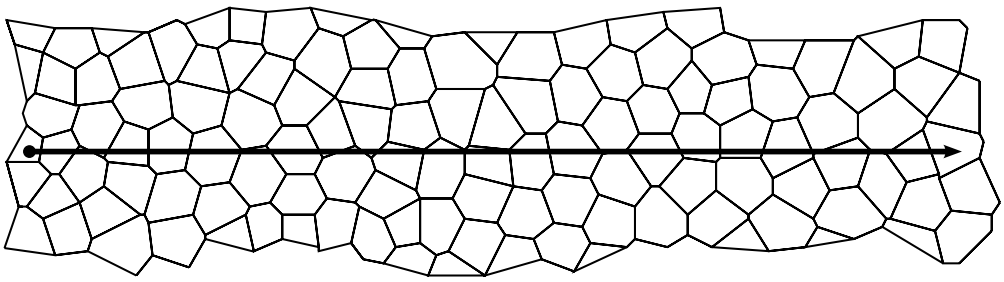

Abb. 7.2: Beispielsituation für Abschätzung der Rechenzeit im Voronoi-Rechengitter

8 Schlussfolgerung und Ausblick

Für das DSMC-Verfahren ist ein Rechengitter besonders geeignet, wenn es aus konvexen Zellen mit etwa gleicher Ausdehnung in verschiedenen Richtungen besteht und wenn die Netzdichte möglichst kontinuierlich verändert werden kann, um das Gitter an die sich verändernde Strömung anzupassen. Die häufig verwendeten Advancing-Front- oder Delaunay-Triangulationen sind keine optimale Lösung, weil sie in der Regel nur Dreiecke bzw. Tetraeder erzeugen, und weil deren Qualität bei einer Adaption schnell abnehmen kann. Der Quadtree wäre mit rechteckigen Zellen schon besser geeignet, wenn sich verhindern ließe, dass an der Randkontur deutlich schlechtere Formen auftreten. Das Voronoi-Rechengitter kommt dem idealen DSMC-Rechengitter eindeutig am nächsten. Es lässt sich leicht Konstruieren und durch Hinzufügen und Entfernen von Zellen verändern. Der größte leere Kreis für die Position neuer Zentren sorgt einerseits für eine guter Form der Zellen und andererseits für eine kontinuierliche Änderung der Netzdichte. Da von der Randkontur nur vorausgesetzt wurde, dass sie aus geschlossenen, nicht überlappenden Polygonen bzw. Polyedern besteht, kann man davon ausgehen, dass das Voronoi-Rechengitter auch für komplizierte, dreidimensionale Geometrien geeignet ist.
Ein ebenso wichtiger Faktor für das DSMC-Verfahren ist die Rechenzeit, die für die Teilchenverfolgung durch das Gitter benötigt wird, weil eine größere Zahl von Simulationspartikeln bessere Ergebnisse ermöglicht. Für den Quadtree wurde eine Teilchenverfolgung angegeben, die in Abwesenheit von Rändern extrem effizient ist. Eigentlich ist der Quadtree schon aufgrund der Möglichkeit beliebig schlecht geformter Zellen für DSMC ungeeignet. Die Teilchenverfolgung zeigt aber, welchen Vorteil ein hierarchisch aufgebautes Gitter haben kann. Deshalb wird weiter unten eine Idee vorgestellt, wie dieser Vorteil auch im Voronoi-Rechengitter ausgenutzt werden kann.
Nach der Adaption des Voronoi-Gitters werden häufig einige Teilchen falsch zugeordnet sein. Nach einer Bewegung aller Teilchen müssen diese aber unbedingt korrekt zugeordnet sein, weil sonst die Ergebnisse verfälscht würden. Deshalb wurde für

die Teilchenverfolgung im Voronoi-Gitter der Algorithmus 1 vorgestellt, der solche Fehlzuordnungen bei der Bewegung korrigiert. Darauf aufbauend konnte der Algorithmus 2 entwickelt werden, der ebenfalls in unstrukturierten Gittern aus konvexen Zellen beliebige Fehler korrigiert und dafür nur etwa die Hälfte der Rechenzeit benötigt.
Der einzige erkennbare Nachteil, das Voronoi-Rechengitter für das DSMC-Verfahren einzusetzen, besteht darin, dass im Gegensatz zu bereits vorhandenen Methoden zur Gittergenerierung keine fertige Implementierung und keine Erfahrungen existieren. Allerdings gibt es Erfahrungen mit Voronoi-Diagrammen in vielen anderen Bereichen die eventuell für das Voronoi-Rechengitter genutzt werden könnten. So gibt es z.B. für die Auswertung der Daten bzw. für den Adaptionsprozess einige Analogien zu der Auswertung geographischer Daten bzw. Untersuchungen zur Dynamik der Aufteilung von Wirtschaftsmärkten (Hotelling-Process), in denen ebenfalls Voronoi-Diagramme eingesetzt werden [16]. Dass sich der zusätzliche Aufwand, ein Voronoi-Rechengitter zu implementieren, lohnen könnte, sollen einige einfache, aber möglicherweise sehr wirkungsvolle Ideen zur Weiterentwicklung des Voronoi-Rechengitters zeigen.

Wiederverwendbare Rechengitter

Das Voronoi-Rechengitter basiert komplett auf lokalen Veränderungen. Deshalb ist es nicht unbedingt notwendig, immer wieder neue Gitter aufzubauen. Anstatt wie beschrieben erst ein minimales Gitter um einen Rand zu erzeugen und dieses zu adaptieren, könnten auch Gitter verschiedener Dichte gespeichert werden und diese an die jeweilige Randkontur angepasst werden. Ein weiterer Vorteil davon wäre, dass dann auch schon an eine Strömung adaptierte Gitter bei einer z.B. durch Ablation leicht veränderten Randkontur weiterverwendet werden können.

Adaption

Die Abbildung 4.4 in Kapitel 4.3 zeigt, dass ein Voronoi-Rechengitter durch Hinzufügen neuer Zentren im größten leeren Umkreis bestehender Zentren auch bei variabler Netzdichte leicht eine hohe Gitterqualität behält. Eine andere Aufgabe ist es aber, diese Netzdichte optimal mit der zeitlich veränderlichen Strömung anzupassen. Optimal bedeutet, dass bei möglichst wenig Veränderungen am Gitter die Netzdichte möglichst nah an der durch Kriterien wie z.B. mittlere freie Weglänge und Dichte der Teilchen vorgegebenen idealen Netzdichte bleibt. Diese zeitliche Adaption lässt sich durch Hinzufügen und Entfernen von Zellen sehr einfach gestalten.
Man stelle sich die Kriterien, die die Netzdichte bestimmen, als Nahrung für die Zel-

len vor. Bekommt eine Zelle über mehrere Zeitschritte einen Überfluss an Nahrung, führt dies zu einer Zellteilung. Das neue Zentrum wird natürlich wieder in die vom Zentrum entfernteste Ecke des Voronoi-Polygons platziert. Mangelt es einer Zelle an Nahrung, beginnt sie kleinere Nachbarzellen zu fressen, bis das Gleichgewicht wieder hergestellt ist.
Das Einfache an dieser Interpretation des Adaptionsprozesses ist, dass beliebige Kriterien für die Netzdichte oder an den Rändern auch Kriterien für die Form der Zellen inklusive Gewichtung eingeführt werden können, ohne dass die Adaption an Komplexität zunimmt. Über einen Parameter kann die Sensitivität des Gitters auf die Strömung festgelegt werden.

Periodische Gitter

Die Voronoi-Zellen der Zentren auf der konvexen Hülle aller Zentren sind unendlich ausgedehnt (vgl. z.B. die vier äußersten Zentren in Abb. 7.1). Für das Voronoi-Rechengitter spielt das keine Rolle, weil dieses zusätzlich durch die Randkontur begrenzt wird. Aber in einigen anderen Anwendungen von Voronoi-Diagrammen [16] verhindert man solche Randeffekte, indem für die Zentren periodische Randbedingungen gelten. Mit diesem Prinzip ergibt sich die interessante Möglichkeit, periodische Randbedingungen in der Teilchenverfolgung durch ein periodisches Gitter zu ersetzen.
In Abbildung 8.1 wurden zunächst nur Zentren zwischen den beiden gestrichelten Linien erzeugt. Die Zentren, deren Voronoi-Polygon eine der beiden Linien schneiden, wurden hinter die gegenüberliegende Linie kopiert. Würde man dieses Voronoi-Diagramm entlang der hervorgehobenen Linien ausschneiden, könnte man das Voronoi-Diagramm auf eine Zylinderoberfläche legen und die beiden Enden würden genau ineinander passen. So lassen sich auch in drei Dimensionen periodische Randbedingungen einführen, ohne dass die Teilchen erst auf einer Seite auf eine Ebene treffen müssen, um dann auf der gegenüberliegenden Seite wieder aufzutauchen.

Hierarchische Gitter

Der wesentliche Vorteil des Quadtrees ist, dass durch die hierarchische Struktur verschiedene Stufen genutzt werden können und die Teilchen nach einer Bewegung für die Auswertung wieder in den selben Zellen liegen.
Auch bei einigen Anwendungen von Voronoi-Diagrammen gibt es eine Hierarchie unter den Zentren [16], wie z.B. Städte auf einer Landkarte als Zentren auf verschiedenen Verwaltungsebenen. In Abbildung 8.2 ist ein hierarchisches Voronoi-Diagramm

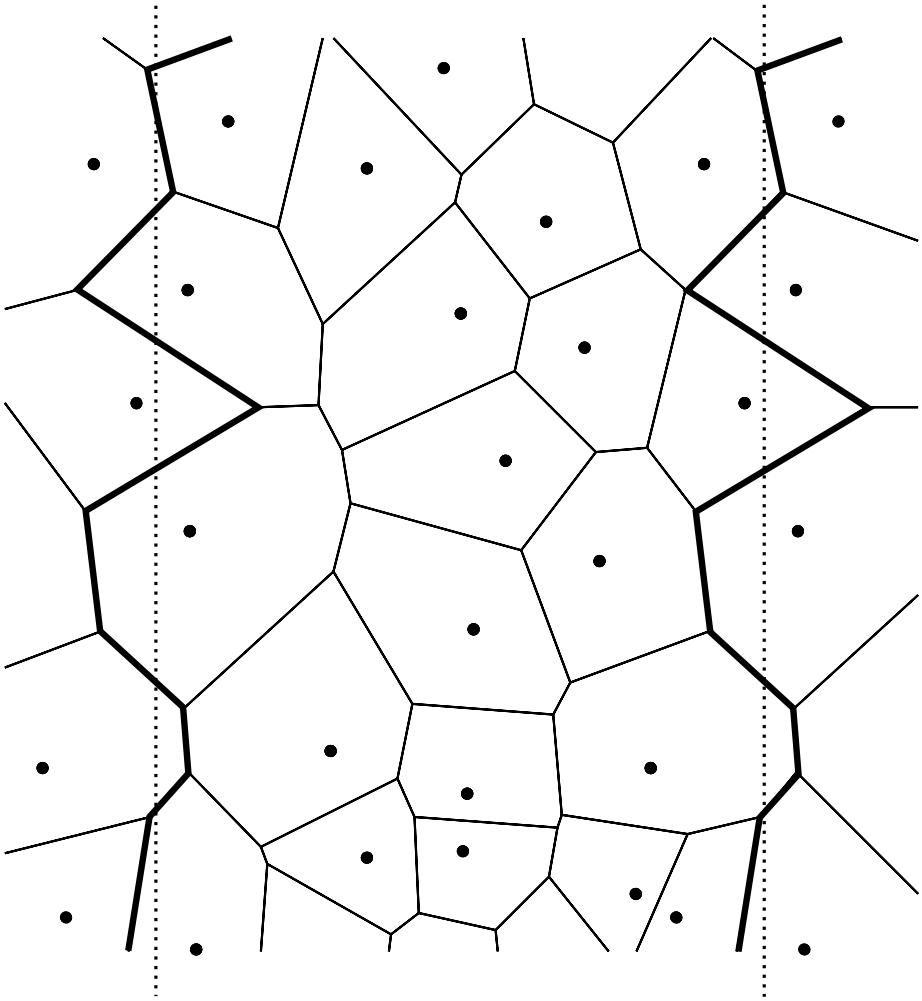

Abb. 8.1: Periodisches Voronoi-Diagramm

aus drei Ebenen zu sehen. Die großen Zentren bilden die erste Ebene und sind in allen tieferen Ebenen enthalten. Die kleineren Zentren sind nur in der zweiten und dritten Ebene enthalten. Dabei wurden die drei Ebenen nicht einzeln erzeugt, sondern eine neue Ebene aus ihrer Vorgängerebene abgeleitet, sodass zu dieser Konstruktion kaum mehr Rechenzeit benötigt wird, als wenn die niedrigste Ebene allein erzeugt worden wäre.

Ein so aufgebautes Rechengitter kann von großen Nutzen sein, wenn sich die Geschwindigkeitsverteilung der Teilchen über einen großen Bereich erstreckt. Teilchen mit Geschwindigkeiten, die deutlich größer sind als die Ausdehnung der Zellen, müssten so nicht die vielen kleinen Zellen durchqueren. Sie könnten durch eine für die Geschwindigkeit geeigneteren Ebene des Gitters verfolgt werden, bis hier die rich-

tige Zelle gefunden wurde. Dann wird das Teilchen ausgehend vom Zentrum dieser Zelle in der nächst niedrigeren Ebene verfolgt, weil genau dieses Zentrum näher am Zielort liegt, als alle anderen Zentren der selben Ebene.

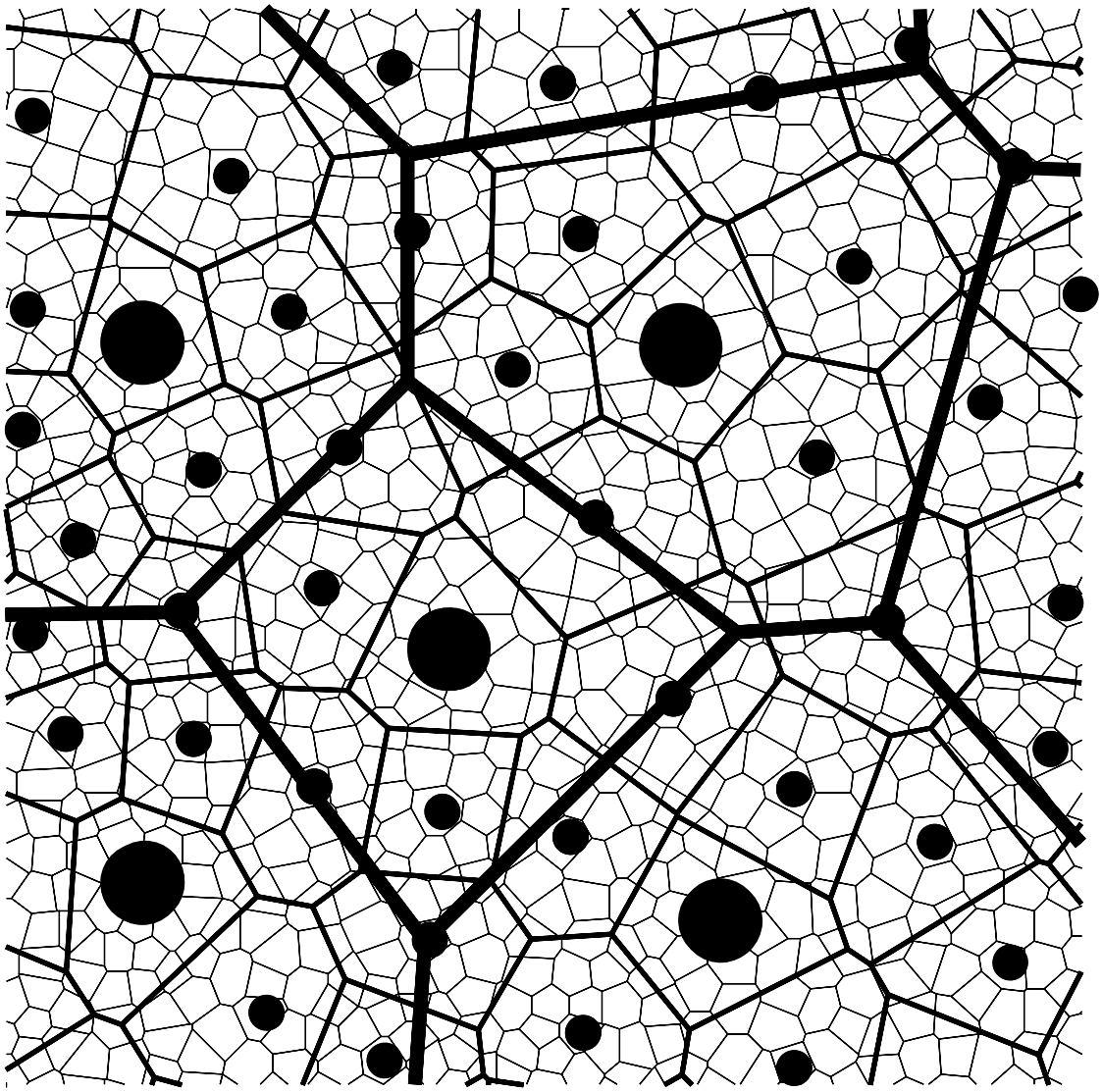

Abb. 8.2: Hierarchisches Voronoi-Diagramm mit dem Anzahlenverhältnis 1:16 von einer Ebene zur nächsten Ebene

Literaturverzeichnis

[1] Aurenhammer, F. ; Klein, R.: Voronoi Diagrams. In: *Handbook of Computational Geometry*, Elsevier Science Publishers B.V. North-Holland, 2000, S. 201–290

[2] Bird, G. A.: *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*. Clarendon Press, 1994

[3] Blacker, T. D. ; Stephenson, M. B.: Paving: A new approach to automated quadrilateral mesh generation. In: *International Journal for numerical methods in engineering* 32 (1991)

[4] Cavalcanti, P.R. ; Mello, U.T.: Three-dimensional constrained Delaunay-Triangulation: a minimalist approach. In: *Proceedings of the 8th International Meshing Roundtable*, 1999, S. 119–129

[5] Devillers, O. ; Teillaud, M.: Perturbations and Vertex Removal in a 3D Delaunay Triangulations. In: *Symposium on Discrete Algorithms* 15 (2003)

[6] Hales, T. C.: The Honeycomb Conjecture. In: *Discrete and Computational Geometry* 25 (2000)

[7] Haselbacher, A.: An efficient and robust particle-localization algorithm for unstrucktured grids. In: *Journal of Computational Physics* 225 (2007)

[8] Hertel, S. ; Mehlhorn, K.: Fast triangulations of simple polygons. In: *Lecture Notes in Computational Science* 158 (1983)

[9] Joe, B.: Construction of three-dimensional Delaunay triangulations using local transformations. In: *Computer Aided Geometric Design* 8 (1991)

[10] Kinney, P.: Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing. In: *Proceedings of the 6th International Meshing Roundtable* (1997)

[11] Klein, R.: *Algorithmische Geometrie*. Springer Verlag, 2005

[12] Laux, M.: *Direkte Simulation verdünnter, reagierender Strömungen*. VDI Verlag, 1996

[13] Löhner, R. ; Ambrosiano, J.: A Voctorized Particle Tracer for Unstructured Grids. In: *Journal of Computational Physics* 91 (1989)

[14] Macrossan, M. N.: Scaling parameters for hypersonic flow: correlation of sphere drag data. In: *25th International Symposium on Rarefied Gas Dynamics* (2006)

[15] Mostafavia, M. A. ; Goldb, C. ; Dakowicz, M.: Delete and insert operations in Voronoi/Delaunay methods and applications. In: *Computers & Geosciences* 29 (2002)

[16] Okabe, A. ; Boot, B. ; Sugihara, K.: *Spatial Tessallations: Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons Ltd., 1992

[17] O'Rourke, J.: *Art Gallary Theorems and Algorithms*. Oxford University Press, 1987

[18] Peraire, J. ; Peiro, J. ; Morgan, K.: Advancing Front Grid Generation. In: *J. F. Thompson (Hrsg.): Handbook of Grid Generation*. CRC Press LLC, 1999, Kap. 17

[19] Ruppert, J.: A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. In: *Journal of Algorithms* 18 (1995)

[20] Sugihara, K. ; Inagaki, H.: Why is the 3D Delaunay triangulation difficult to construct? In: *Information Processing Letters* 54 (1995)

Erklärung nach §13(8)der Prüfungsordnung für den Bachelor-Studiengang Physik und den Master-Studiengang Physik an der Universität Göttingen:

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.
Darüberhinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, im Rahmen einer nichtbestandenen Prüfung an dieser oder einer anderen Hochschule eingereicht wurde.

Göttingen, den 9. August 2009

(Frank Stollmeier)